

68

MICRO JOURNAL

Australia A \$ 4.75 New Zealand NZ \$ 6.50
 Singapore S \$ 9.45 Hong Kong H \$ 23.50
 Malaysia M \$ 9.45 Sweden 30:-SEK

\$2.95_{USA}

OS-9 Atari Amiga Mac S-50

6800 6809 68008 68000 68010 68020 68030

The Magazine for Motorola CPU Devices For Over a Decade!

This Issue: "C" User Notes p.6

Motorola 88000 RISC p. 25

Logically Speaking p.13

Mac Watch p.41

Position Independent & Reentrant Code for
 the 68000 Family p. 45

OS-9 SE*DOS Atari Amiga
 FLEX Macintosh

A User Contributor Journal

And Lots More!

VOLUME XI ISSUE II • Devoted to the 68XXX User • February 1989

The Grandfather of "DeskTop Publishing™"

SERVING THE 68XXX USER WORLDWIDE

000422 A/E
 MR. MICKEY FERGUSON
 P.O. BOX 87
 KINGSTON SPRINGS TN 37082

MJ

MC 6809E

MC 68008

MC 6809 CPU

A0 D0
 A1 D1
 A2 D2
 A3 D3
 A4 D4
 A5 D5
 A6 D6
 A7 D7
 A8 D8
 A9 D9
 A10 D10
 A11 D11
 A12 D12
 A13 D13
 A14 D14
 A15 D15
 A16 D16
 A17 D17
 A18 D18
 A19 D19
 A20 D20
 A21 D21
 A22 D22
 A23 D23
 A24 D24
 A25 D25
 A26 D26
 A27 D27
 A28 D28
 A29 D29
 A30 D30
 A31 D31
 A32 D32

D40
 D41
 D42
 D43
 D44
 D45
 D46
 D47
 D48
 D49
 D50
 D51
 D52
 D53
 D54
 D55
 D56
 D57
 D58
 D59
 D60
 D61
 D62
 D63
 D64
 D65
 D66
 D67
 D68
 D69
 D70
 D71
 D72
 D73
 D74
 D75
 D76
 D77
 D78
 D79
 D80
 D81
 D82
 D83
 D84
 D85
 D86
 D87
 D88
 D89
 D90
 D91
 D92
 D93
 D94
 D95
 D96
 D97
 D98
 D99
 D100
 D101
 D102
 D103
 D104
 D105
 D106
 D107
 D108
 D109
 D110
 D111
 D112
 D113
 D114
 D115
 D116
 D117
 D118
 D119
 D120
 D121
 D122
 D123
 D124
 D125
 D126
 D127
 D128
 D129
 D130
 D131
 D132
 D133
 D134
 D135
 D136
 D137
 D138
 D139
 D140
 D141
 D142
 D143
 D144
 D145
 D146
 D147
 D148
 D149
 D150
 D151
 D152
 D153
 D154
 D155
 D156
 D157
 D158
 D159
 D160
 D161
 D162
 D163
 D164
 D165
 D166
 D167
 D168
 D169
 D170
 D171
 D172
 D173
 D174
 D175
 D176
 D177
 D178
 D179
 D180
 D181
 D182
 D183
 D184
 D185
 D186
 D187
 D188
 D189
 D190
 D191
 D192
 D193
 D194
 D195
 D196
 D197
 D198
 D199
 D200
 D201
 D202
 D203
 D204
 D205
 D206
 D207
 D208
 D209
 D210
 D211
 D212
 D213
 D214
 D215
 D216
 D217
 D218
 D219
 D220
 D221
 D222
 D223
 D224
 D225
 D226
 D227
 D228
 D229
 D230
 D231
 D232
 D233
 D234
 D235
 D236
 D237
 D238
 D239
 D240
 D241
 D242
 D243
 D244
 D245
 D246
 D247
 D248
 D249
 D250
 D251
 D252
 D253
 D254
 D255
 D256
 D257
 D258
 D259
 D260
 D261
 D262
 D263
 D264
 D265
 D266
 D267
 D268
 D269
 D270
 D271
 D272
 D273
 D274
 D275
 D276
 D277
 D278
 D279
 D280
 D281
 D282
 D283
 D284
 D285
 D286
 D287
 D288
 D289
 D290
 D291
 D292
 D293
 D294
 D295
 D296
 D297
 D298
 D299
 D300
 D301
 D302
 D303
 D304
 D305
 D306
 D307
 D308
 D309
 D310
 D311
 D312
 D313
 D314
 D315
 D316
 D317
 D318
 D319
 D320
 D321
 D322
 D323
 D324
 D325
 D326
 D327
 D328
 D329
 D330
 D331
 D332
 D333
 D334
 D335
 D336
 D337
 D338
 D339
 D340
 D341
 D342
 D343
 D344
 D345
 D346
 D347
 D348
 D349
 D350
 D351
 D352
 D353
 D354
 D355
 D356
 D357
 D358
 D359
 D360
 D361
 D362
 D363
 D364
 D365
 D366
 D367
 D368
 D369
 D370
 D371
 D372
 D373
 D374
 D375
 D376
 D377
 D378
 D379
 D380
 D381
 D382
 D383
 D384
 D385
 D386
 D387
 D388
 D389
 D390
 D391
 D392
 D393
 D394
 D395
 D396
 D397
 D398
 D399
 D400
 D401
 D402
 D403
 D404
 D405
 D406
 D407
 D408
 D409
 D410
 D411
 D412
 D413
 D414
 D415
 D416
 D417
 D418
 D419
 D420
 D421
 D422
 D423
 D424
 D425
 D426
 D427
 D428
 D429
 D430
 D431
 D432
 D433
 D434
 D435
 D436
 D437
 D438
 D439
 D440
 D441
 D442
 D443
 D444
 D445
 D446
 D447
 D448
 D449
 D450
 D451
 D452
 D453
 D454
 D455
 D456
 D457
 D458
 D459
 D460
 D461
 D462
 D463
 D464
 D465
 D466
 D467
 D468
 D469
 D470
 D471
 D472
 D473
 D474
 D475
 D476
 D477
 D478
 D479
 D480
 D481
 D482
 D483
 D484
 D485
 D486
 D487
 D488
 D489
 D490
 D491
 D492
 D493
 D494
 D495
 D496
 D497
 D498
 D499
 D500
 D501
 D502
 D503
 D504
 D505
 D506
 D507
 D508
 D509
 D510
 D511
 D512
 D513
 D514
 D515
 D516
 D517
 D518
 D519
 D520
 D521
 D522
 D523
 D524
 D525
 D526
 D527
 D528
 D529
 D530
 D531
 D532
 D533
 D534
 D535
 D536
 D537
 D538
 D539
 D540
 D541
 D542
 D543
 D544
 D545
 D546
 D547
 D548
 D549
 D550
 D551
 D552
 D553
 D554
 D555
 D556
 D557
 D558
 D559
 D560
 D561
 D562
 D563
 D564
 D565
 D566
 D567
 D568
 D569
 D570
 D571
 D572
 D573
 D574
 D575
 D576
 D577
 D578
 D579
 D580
 D581
 D582
 D583
 D584
 D585
 D586
 D587
 D588
 D589
 D590
 D591
 D592
 D593
 D594
 D595
 D596
 D597
 D598
 D599
 D600
 D601
 D602
 D603
 D604
 D605
 D606
 D607
 D608
 D609
 D610
 D611
 D612
 D613
 D614
 D615
 D616
 D617
 D618
 D619
 D620
 D621
 D622
 D623
 D624
 D625
 D626
 D627
 D628
 D629
 D630
 D631
 D632
 D633
 D634
 D635
 D636
 D637
 D638
 D639
 D640
 D641
 D642
 D643
 D644
 D645
 D646
 D647
 D648
 D649
 D650
 D651
 D652
 D653
 D654
 D655
 D656
 D657
 D658
 D659
 D660
 D661
 D662
 D663
 D664
 D665
 D666
 D667
 D668
 D669
 D670
 D671
 D672
 D673
 D674
 D675
 D676
 D677
 D678
 D679
 D680
 D681
 D682
 D683
 D684
 D685
 D686
 D687
 D688
 D689
 D690
 D691
 D692
 D693
 D694
 D695
 D696
 D697
 D698
 D699
 D700
 D701
 D702
 D703
 D704
 D705
 D706
 D707
 D708
 D709
 D710
 D711
 D712
 D713
 D714
 D715
 D716
 D717
 D718
 D719
 D720
 D721
 D722
 D723
 D724
 D725
 D726
 D727
 D728
 D729
 D730
 D731
 D732
 D733
 D734
 D735
 D736
 D737
 D738
 D739
 D740
 D741
 D742
 D743
 D744
 D745
 D746
 D747
 D748
 D749
 D750
 D751
 D752
 D753
 D754
 D755
 D756
 D757
 D758
 D759
 D760
 D761
 D762
 D763
 D764
 D765
 D766
 D767
 D768
 D769
 D770
 D771
 D772
 D773
 D774
 D775
 D776
 D777
 D778
 D779
 D780
 D781
 D782
 D783
 D784
 D785
 D786
 D787
 D788
 D789
 D790
 D791
 D792
 D793
 D794
 D795
 D796
 D797
 D798
 D799
 D800
 D801
 D802
 D803
 D804
 D805
 D806
 D807
 D808
 D809
 D810
 D811
 D812
 D813
 D814
 D815
 D816
 D817
 D818
 D819
 D820
 D821
 D822
 D823
 D824
 D825
 D826
 D827
 D828
 D829
 D830
 D831
 D832
 D833
 D834
 D835
 D836
 D837
 D838
 D839
 D840
 D841
 D842
 D843
 D844
 D845
 D846
 D847
 D848
 D849
 D850
 D851
 D852
 D853
 D854
 D855
 D856
 D857
 D858
 D859
 D860
 D861
 D862
 D863
 D864
 D865
 D866
 D867
 D868
 D869
 D870
 D871
 D872
 D873
 D874
 D875
 D876
 D877
 D878
 D879
 D880
 D881
 D882
 D883
 D884
 D885
 D886
 D887
 D888
 D889
 D890
 D891
 D892
 D893
 D894
 D895
 D896
 D897
 D898
 D899
 D900
 D901
 D902
 D903
 D904
 D905
 D906
 D907
 D908
 D909
 D910
 D911
 D912
 D913
 D914
 D915
 D916
 D917
 D918
 D919
 D920
 D921
 D922
 D923
 D924
 D925
 D926
 D927
 D928
 D929
 D930
 D931
 D932
 D933
 D934
 D935
 D936
 D937
 D938
 D939
 D940
 D941
 D942
 D943
 D944
 D945
 D946
 D947
 D948
 D949
 D950
 D951
 D952
 D953
 D954
 D955
 D956
 D957
 D958
 D959
 D960
 D961
 D962
 D963
 D964
 D965
 D966
 D967
 D968
 D969
 D970
 D971
 D972
 D973
 D974
 D975
 D976
 D977
 D978
 D979
 D980
 D981
 D982
 D983
 D984
 D985
 D986
 D987
 D988
 D989
 D990
 D991
 D992
 D993
 D994
 D995
 D996
 D997
 D998
 D999
 D1000
 D1001
 D1002
 D1003
 D1004
 D1005
 D1006
 D1007
 D1008
 D1009
 D1010
 D1011
 D1012
 D1013
 D1014
 D1015
 D1016
 D1017
 D1018
 D1019
 D1020
 D1021
 D1022
 D1023
 D1024
 D1025
 D1026
 D1027
 D1028
 D1029
 D1030
 D1031
 D1032
 D1033
 D1034
 D1035
 D1036
 D1037
 D1038
 D1039
 D1040
 D1041
 D1042
 D1043
 D1044
 D1045
 D1046
 D1047
 D1048
 D1049
 D1050
 D1051
 D1052
 D1053
 D1054
 D1055
 D1056
 D1057
 D1058
 D1059
 D1060
 D1061
 D1062
 D1063
 D1064
 D1065
 D1066
 D1067
 D1068
 D1069
 D1070
 D1071
 D1072
 D1073
 D1074
 D1075
 D1076
 D1077
 D1078
 D1079
 D1080
 D1081
 D1082
 D1083
 D1084
 D1085
 D1086
 D1087
 D1088
 D1089
 D1090
 D1091
 D1092
 D1093
 D1094
 D1095
 D1096
 D1097
 D1098
 D1099
 D1100
 D1101
 D1102
 D1103
 D1104
 D1105
 D1106
 D1107
 D1108
 D1109
 D1110
 D1111
 D1112
 D1113
 D1114
 D1115
 D1116
 D1117
 D1118
 D1119
 D1120
 D1121
 D1122
 D1123
 D1124
 D1125
 D1126
 D1127
 D1128
 D1129
 D1130
 D1131
 D1132
 D1133
 D1134
 D1135
 D1136
 D1137
 D1138
 D1139
 D1140
 D1141
 D1142
 D1143
 D1144
 D1145
 D1146
 D1147
 D1148
 D1149
 D1150
 D1151
 D1152
 D1153
 D1154
 D1155
 D1156
 D1157
 D1158
 D1159
 D1160
 D1161
 D1162
 D1163
 D1164
 D1165
 D1166
 D1167
 D1168
 D1169
 D1170
 D1171
 D1172
 D1173
 D1174
 D1175
 D1176
 D1177
 D1178
 D1179
 D1180
 D1181
 D1182
 D1183
 D1184
 D1185
 D1186
 D1187
 D1188
 D1189
 D1190
 D1191
 D1192
 D1193
 D1194
 D1195
 D1196
 D1197
 D1198
 D1199
 D1200
 D1201
 D1202
 D1203
 D1204
 D1205
 D1206
 D1207
 D1208
 D1209
 D1210
 D1211
 D1212
 D1213
 D1214
 D1215
 D1216
 D1217
 D1218
 D1219
 D1220
 D1221
 D1222
 D1223
 D1224
 D1225
 D1226
 D1227
 D1228
 D1229
 D1230
 D1231
 D1232
 D1233
 D1234
 D1235
 D1236
 D1237
 D1238
 D1239
 D1240
 D1241
 D1242
 D1243
 D1244
 D1245
 D1246
 D1247
 D1248
 D1249
 D1250
 D1251
 D1252
 D1253
 D1254
 D1255
 D1256
 D1257
 D1258
 D1259
 D1260
 D1261
 D1262
 D1263
 D1264
 D1265
 D1266
 D1267
 D1268
 D1269
 D1270
 D1271
 D1272
 D1273
 D1274
 D1275
 D1276
 D1277
 D1278
 D1279
 D1280
 D1281
 D1282
 D1283
 D1284
 D1285
 D1286
 D1287
 D1288
 D1289
 D1290
 D1291
 D1292
 D1293
 D1294
 D1295
 D1296
 D1297
 D1298
 D1299
 D1300
 D1301
 D1302
 D1303
 D1304
 D1305
 D1306
 D1307
 D1308
 D1309
 D1310
 D1311
 D1312
 D1313
 D1314
 D1315
 D1316
 D1317
 D1318
 D1319
 D1320
 D1321
 D1322
 D1323
 D1324
 D1325
 D1326
 D1327
 D1328
 D1329
 D1330
 D1331
 D1332
 D1333
 D1334
 D1335
 D1336
 D1337
 D1338
 D1339
 D1340
 D1341
 D1342
 D1343
 D1344
 D1345
 D1346
 D1347
 D1348
 D1349
 D1350
 D1351
 D1352
 D1353
 D1354
 D1355
 D1356
 D1357
 D1358
 D1359

WHO DO YOU CALL WHEN YOUR DEBUGGER WON'T DEBUG?



The problem with most real-time operating systems is simple, they're not an integrated solution. You end up dealing with a multitude of suppliers for languages, compilers, debuggers and other important development tools. And when something does go wrong, it can be a frustrating experience trying to straighten out the mess.

Why Not Try the Microware One-Stop Total Solution?

Microware's OS-9 Real-Time Operating System is a total integrated software system, not just a kernel. We offer an extensive set of development tools, languages, I/O and Kernel options. *And this total integrated solution is entirely designed, built and supported by the same expert Microware team.*

Microware is a registered trademark of Microware Systems Corporation.
OS-9 is a trademark of Microware.
UNIX is a trademark of AT&T.
VAX is a trademark of DEC.

Modularity Lets YOU Choose Just What You Need.

The modular design of OS-9 allows our Operating System to adapt as your requirements change. OS-9 can support a complete spectrum of applications — from embedded ROM-based code in board-level products all the way up to large-scale systems.

Support is Part of the Package.

Microware is proudly setting the industry's standard for customer support. You'll find professional and comprehensive technical documentation and a Customer Hotline staffed by courteous and authoritative software engineers.

So stop messing with simple kernels and independent suppliers. Call Microware today and find out more about the "One-Stop Integrated Solution" with OS-9!

The OS-9 Success Kit

A Total Integrated Solution for Your Next Project

Development Tools:

C Source Level Debugger
Symbolic Debugger
System State Debugger
uMACS Text Editor
Electronic Mail
Communications
Super Shell

Kernel Options:

MMU (Security Protection) Support
Math Coprocessor Support

* Resident or UNIX versions available
** VAX hosted

Languages:

C*
Basic
Pascal
Fortran
Ada**
Assembler*

I/O Options:

SCSI, SASI & SMD Disks
3-, 5-, 8-inch Diskettes
Magnetic Tape
Ethernet - TCP/IP
Arcnet - OS-9/Net

microware® **OS-9**

Microware Systems Corporation
1900 N.W. 114th Street
Des Moines, Iowa 50322
Phone: 515/224-1929

Western Regional Office
4401 Great America Parkway
Santa Clara, California 95054
Phone: 408/980-0201

Microware Japan Ltd.
41-19 Honcho 4-Chome
Funabashi City
Chiba 273, Japan
Phone: 0474 (22) 1747

Mustang-020 Mustang-08 Benchmarks

IBM AT 7300 Xerox Sys 3
 AT&T 7300 UNIX PC 68010
 DEC VAX 11/780 UNIX Berkeley 4.2
 DEC VAX 11/750
 68008 OS-9 68K 8 Mhz
 68000 OS-9 68K 10 Mhz
 MUSTANG-08 68008 OS-9 68K 10 Mhz
 MUSTANG-020 68020 OS-9 68K 16 Mhz
 MUSTANG-020 68020 MC68881 UniFLEX 16 Mhz

32 bit Integer	Registers Long
9.7	
7.2	4.3
3.6	3.2
5.1	3.2
18.0	9.0
6.5	4.0
9.8	6.3
2.2	0.88
1.8	1.22

Main()

register long i;
 for (i=0; i < 999999; ++i);

Estimated MIPS - MUSTANG-020 ... 4.5 MIPS,
 Burst to 8 - 10 MIPS; Motorola Specs

OS-9

OS-9 Professional Ver	\$850.00
*Includes C Compiler	
Basic OS	300.00
C Compiler	500.00
68000 Disassembler (w/source add: \$100.00)	100.00
Fortran 77	750.00
Microvare Pascal	500.00
OrangeSoft Pascal	900.00
Style-Graph	495.00
Style-Spell	195.00
Style-Merge	175.00
Style-Graph-Spell-Merge	695.00
PAT w/C source	239.00
JUST w/C source	79.95
PAT/JUST Combo	249.50
Scalysur + (see below)	995.00
COM	125.00

UniFLEX

UniFLEX (68020 ver)	\$450.00
Screen Editor	150.00
Sort-Merge	200.00
BASIC/Pro Compiler	300.00
C Compiler	350.00
COBOL	750.00
CMODEM w/source	100.00
TMODEM w/source	100.00
X-TALK (see Ad)	99.95
Cross Assembler	50.00
Fortran 77	450.00
Scalysur + (see below)	995.00

Standard MUSTANG-020 shipped 12.5 Mhz.
 Add for 16.6 Mhz 68020 375.00
 Add for 16.6 Mhz 68881 375.00
 Add for 20 Mhz 68020/RAM 750.00

16 Pin exp. RS-232 335.00

Requires 1 or 2 Adapter Cards below RS232 Adapter 165.00
 Each card supports 4 additional ser. ports
 (total of 36 serial ports supported)

60 line Parallel I/O card 398.00
 Uses 3 68230 Interface/Terminal chips,
 6 groups of 8 lines each, separate buffer
 direction control for each group.

Prototype Board 75.00
 areas for both dip and PGA devices & a
 pre-wired memory area up to 512K DRAM.

SBC-AN 475.00
 Interface between the system and
 ARCNET modified token-passing LAN, fiber optic optional - call.
 LAN software drivers 120.00

Expansion for Motorola I/O Channel Modules \$195.00
 Special for complete MUSTANG-020 system buyers - Scalysur+
 \$695.00. SAVE \$300.00
 Software Discounts

All MUSTANG-020 system and board buyers are entitled to
 discounts on all listed software: 10-70% depending on item. Call or
 write for quotes. Discounts apply after the sale as well.

Note: Only Professional OS-9 Now Available
 (68020 Version) Includes (\$500) C Compiler -
 68020 & 68881 Supported - For UPGRADES
 Write or Call for Professional OS-9 Upgrade Kit

Mustang Specifications

12.5 Mhz (optional 16.6 Mhz available) MC68020 full 32-bit wide path
 32-bit wide data and address buses, on-chip instruction cache
 on-chip instruction cache
 object code compatible with all 68XXX family processors
 enhanced instruction set - math co-processor interface
 68881 math hi-speed floating point co-processor (optional)
 direct extension of full 68020 instruction set
 full support IEEE P754, draft 10.0
 transcendental and other scientific math functions
 2 Megabyte of SIP RAM (512 x 32 bit organization)
 up to 256K bytes of EPROM (64 x 32 bits)
 4 Asynchronous serial I/O ports standard
 optional 20 serial ports
 standard RS-232 interface
 optional network interface
 buffered 8 bit parallel port (1/2 MC68230)
 Centronics type pinout
 expansion connector for I/O devices
 16 bit data path
 256 byte address space
 2 interrupt inputs
 clock and control signals
 Motorola I/O Channel Modules
 time of day clock/calendar w/battery backup
 controller for 2.5 1/4" floppy disk drives
 single or double side, single or double density
 35 to 80 track selectable (48-96 TPI)
 SASI interface
 programmable periodic interrupt generator
 interrupt rate from micro-seconds to seconds
 highly accurate time base (5 PPM)
 5 bit sense switch, readable by the CPU
 Hardware single-step capability



Don't be misled!
ONLY Data-Comp
 delivers the Super
MUSTANG-020

These hi-speed 68020 systems are presently working at NASA, Atomic Energy Commission,
 Government Agencies as well as Universities, Business, Labs, and other Critical Applications
 Centers, worldwide, where speed, math crunching and multi-user, multi-tasking UNIX C level
 V compatibility and low cost is a must.

Only the "PRO" Version
 of OS-9 Supported!



This is **HEAVY DUTY**
 Country!

For a limited time we will offer a \$400 trade-in on your
 old 68XXX SBC. Must be working properly and
 complete with all software, cables and documentation.
 Call for more information

Price List:	
Mustang-020 SBC	\$2490.00
Cabinet w/switching PS	\$299.95
5"-80 track floppy DS/DD	\$269.95
Floppy Cable	\$39.95
OS-9 68K Professional Version	\$850.00
C Compiler (\$500 Value)	N/C
Winchester Cable	\$39.95
Winchester Drive 25 Mbyte	\$895.00
Hard Disk Controller	\$395.00
Shipping USA UPS	\$20.00
UniFLEX	Less \$100.00
MC68881 f/p math processor	Add \$275.00
16.67 Mhz MC68020	\$375.00
16.67 Mhz MC68881	\$375.00
20 Mhz MC68020 Sys	\$750.00
Note all 68881 chips work with 20 Mhz Sys	
Total:	\$5299.80

NEW LOWER PRICES
 25 Mbyte HD ~~\$4299.80~~ \$3749.80
 85 Mbyte HD ~~\$5748.80~~ \$4548.80

Data-Comp Division



A Decade of Quality Service
 Systems World-Wide

Computer Publishing, Inc. 5900 Cassandra Smith Road
 Telephone 615 842-4601 - Telex 510 600-6630 Hixson, Tn 37343

A Member of the CPI Family

68 Micro Journal

10 Years of Dedication to Motorola CPU Users

6800 6809 68000 68010 68020

The Originator of "DeskTop Publishing™"

Publisher
Don Williams Sr.

Executive Editor
Larry Williams

Production Manager
Tom Williams

Office Manager
Joyce Williams

Subscriptions
Cheryl Hodge

Contributing & Associate Editors

Ron Anderson	Dr. E.M. "Bud" Pass
Ron Voigts	Art Weller
Doug Lurie	Dr. Theo Elbert
Ed Law	& Hundreds More of Us

Contents

"C" User Notes	6	Pass
Software User Notes	13	Anderson
Logically Speaking	18	Jones
Interfacing the 88000	25	Lawell & Quan
Mac-Watch	41	Law
Writing Position Independent & Reentrant Code for the 68000 Family	45	Van Sickle
Bit Bucket	51	
Classifieds	55	

68 MICRO JOURNAL

"Contribute Nothing - Expect Nothing" DMW 1986

COMPUTER PUBLISHING, INC.

"Over a Decade of Service"



68 MICRO JOURNAL
Computer Publishing Center
5900 Cassandra Smith Road
PO Box 849
Hixson, TN 37343

Phone (615) 842-4600 Telex 510 600-6630

Copyrighted © 1987 by Computer Publishing, Inc.

68 Micro Journal is the *original* "DeskTop Publishing" product and has continuously published since 1978 using only micro-computers and special "DeskTop" software. Using first a kit built 6800 micro-computer, a modified "ball" typewriter, and "home grown" DeskTop Publishing software. None was commercially available at that time. For over 10 years we have been doing "DeskTop Publishing"! *We originated what has become traditional "DeskTop Publishing"!* Today 68 Micro Journal is acknowledged as the "Grandfather" of "DeskTop Publishing" technology.

68 Micro Journal (ISSN 0194-5025) is published 12 times a year by Computer Publishing Inc. Second Class Postage paid at Hixson, TN. and additional entries. POSTMASTER: send address changes to 68 Micro Journal, POB 849, Hixson, TN 37343.

Subscription Rates

1 Year \$24.50 USA, Canada & Mexico \$34.00 a year.
Others add \$12.00 a year surface, \$48.00 a year Airmail, USA funds. 2 years \$42.50, 3 years \$64.50 plus additional postage for each additional year.

Items or Articles for Publication

Articles submitted for publication must include authors name, address, telephone number, date and a statement that the material is original and the property of the author. Articles submitted should be on diskette, OS-9, SK-DOS, FLEX, Macintosh or MS-DOS. All printed items should be dark type and satisfactory for photo-reproduction. No blue ink! No hand written articles - please! Diagrams o.k.

Please, do not format with spaces any text indents, charts, etc. (source listing o.k.). We will edit in all formatting. Text should fall flush left and use a carriage return only to indicate a paragraph end. Please write for free authors guide.

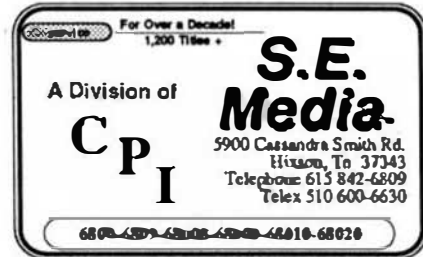
Letters & Advertising Copy

Letters to the Editor should be the original copy, signed! Letters of grip as well as praise are acceptable. *We reserve the right to reject any letter or advertising material, for any reason we deem advisable.* Advertising Rates: Commercial please contact 68 Micro Journal Advertising Department. Classified advertising must be non-commercial. Minimum of \$15.50 for first 15 words. Add \$.60 per word thereafter. No classifieds accepted by telephone.

PAT - JUST

PAT
With 'C' Source
\$229.00

All OS-9
68XXX
Systems



PAT FROM S. E. MEDIA -- A FULL FEATURED SCREEN ORIENTED TEXT EDITOR with all the best of PIE. For those who swore by and loved PIE, this is for YOU! All PIE features & much more! Too many features to list. And if you don't like ours, change or add your own. C source included. Easily configured to your CRT terminal, with special configuration section. No sweat!

68008 - 68000 - 68010 - 68020 OS-9 68K \$229.00

COMBO --- ***PAT/JUST***

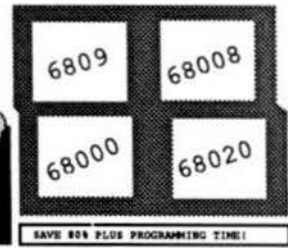
Special \$249.00

JUST

JUST from S. E. MEDIA -- Text formatter written by Ron Anderson; for dot matrix printers, provides many unique features. Output formatted to the display. User configurable for adapting to other printers. Comes set-up for Epson MX80 with Graflex. Up to 10 imbedded printer control commands. Compensates for double width printing. Includes normal line width, page numbering, margin, indent, paragraph, space, vertical skip lines, page length, centering, fill, justification, etc. Use with **PAT** or any other text editor. The **ONLY** stand alone text processor for the 68XXX OS-9 68K, that we have seen. And at a very **LOW PRICE!** Order from: S.E. MEDIA - see catalog this issue.

68008 - 68000 - 68010 - 68020 OS-9 68K
With 'C' source \$79.95

CLOSE OUT SPECIAL SCULPTOR



**From the world's oldest
& largest OS-9 software house!**

**CUTS PROGRAMMING TIME UP TO 80%
6809/68000-68030 **Save 90%****

SCULPTOR-a 4GL - Only from S.E. Media at these prices. OS-9 levels one and two (three GIMIX) 6809, all 68XXX OS-9 standard systems. Regular SCULPTOR versions 1.4:6. One of if not the most efficient and easy to develop DBMS type systems running under OS-9! A system of flexible keyed file access that allows extremely fast record and data retrieval, insertion and deletion or other programmed modifications. Access by key or in ascending order, very fast. The system provides automatic menu generation, compilation and report generation. Practically unlimited custom input format and report formatting. A rich set of maintenance and repair utilities. An extremely efficient development environment that cuts most programming approximately 80% in development and debugging! Portable, at source level, to MS-DOS, UNIX and many other languages and systems.

Standard Version: 1.6 6809 - \$1295.00
68000 \$1295.00
68020 \$1990.00

**Due to a "Special One Time" Purchase, We Are
Making This Savings Offer. Quantities Limited!
*Once this supply is gone the price goes back up!***

System OS-9: 6809/68000-68030

• Regular ~~\$1295.00~~

ONLY

\$99.95

S.E. MEDIA

POB 849 5900 CASSANDRA SMITH ROAD
HIXSON, TN 37343 615 842-4601
TELEX 510 600-6630



SAVE - WHILE SUPPLIES LAST!

Telephone: (615) 842-4600

South East Media

OS-9, UniFLEX, FLEX, SK'DOS SOFTWARE

Telex: 5106006630

!!! Please Specify Your Operating System and Disk Size !!!

SCULPTOR

Full OEM & Dealer Discounts Available!

THE SCULPTOR SYSTEM

Sculptor combines a powerful fourth-generation language with an efficient database management system. Programmers currently using traditional languages such as Basic and Cobol will be amazed at what Sculptor does to their productivity. With Sculptor you'll find that what used to take a week can be achieved in just a few hours.

AN ESTABLISHED LEADER

Sculptor was developed by professionals who needed a software development tool with capabilities that were not available in the software market. It was launched in 1981 and since then, with feedback from an ever-increasing customer base, Sculptor has been refined and enhanced to become one of the most adaptable, fast, and above all reliable systems on the market today.

SYSTEM INDEPENDENCE

Sculptor is available on many different machines and for most operating systems, including MS-DOS, Unix/Xenix and VMS. The extensive list of supported hardware ranges from small personal computers, through multi-user minis up to large mainframes. Sculptor is constantly being ported to new systems.

APPLICATION PORTABILITY

Mobility of software between different environments is one of Sculptor's major advantages. You can develop applications on a stand-alone PC and - without any alterations to the programs - run them on a large multi-user system. For software writers this means that their products can reach a wider marketplace than ever before. It is this system portability, together with high-speed development, that makes Sculptor so appealing to value added resellers, hardware manufacturers and software developers of all kinds.

SPEED AND EFFICIENCY

Sculptor uses a fast and proven indexing technique which provides instant retrieval of data from even the largest of files. Sculptor's fourth generation language is compiled to a compact intermediate code which executes with impressive speed.

INTERNATIONALLY ACCEPTED

By using a simple configuration utility, Sculptor can present information in the language and format that you require. This makes it an ideal product for software development almost anywhere in the world. Australia, the Americas and Europe - Sculptor is already at work in over 20 countries.

THE PACKAGE

With every development system you receive:

- ☐ A manual that makes sense
- ☐ A periodic newsletter
- ☐ Screen form language
- ☐ Report generator
- ☐ Menu system
- ☐ Query facility
- ☐ Set of utility programs
- ☐ Sample programs

For resale products, the run-time system is available at a nominal cost.

Facts

Features

DATA DICTIONARY

Each file may have one or more record types described. Fields may have a name, heading, type, size, format and validation list. Field type may be chosen from:

- ☐ alphanumeric
- ☐ integer
- ☐ floating point
- ☐ money
- ☐ date

DATA FILE STRUCTURE

- ☐ Pictorial, fixed-length records
- ☐ Money stored in lower currency unit
- ☐ Dates stored as integer day numbers

INDEXING TECHNIQUE

Sculptor maintains a B-tree index for each data file. Program logic allows any numbers of alternative indexes to be coded into one other file.

INPUT DATA VALIDATION

Input data may be validated at three levels:

- ☐ automatic by field type
- ☐ validation list in data dictionary
- ☐ programmer coded logic

ARITHMETIC OPERATORS

- Unary minus
- * Multiplication
- / Division
- % Remainder
- + Addition
- Subtraction

MAXIMA AND MINIMA

- Minimum key length 1 byte
- Maximum key length 160 bytes
- Minimum record length 3 bytes
- Maximum record length 32767 bytes
- Maximum fields per record 32767
- Maximum records per file 18 million
- Maximum files per program 16
- Operating system limits

PROGRAMS

- ☐ Define record layout
- ☐ Create new indexed file
- ☐ Generate standard screen-form program
- ☐ Generate standard report program
- ☐ Compile screen-form program
- ☐ Compile report program
- ☐ Screen-form program interpreter
- ☐ Report program interpreter
- ☐ Menu interpreter

RELATIONAL OPERATORS

- = Equal to
- < Less than
- > Greater than
- <= Less than or equal to
- >= Greater than or equal to
- <> Not equal to
- and Logical and
- or Logical or
- between Contains
- begin with Begins with

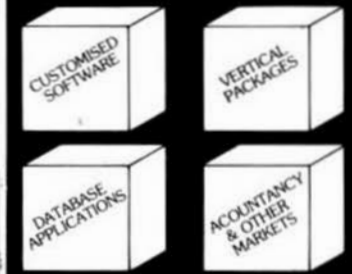
SPECIAL FEATURES

- ☐ Full date arithmetic
- ☐ Echo suppression for passwords
- ☐ Terminal and printer independence
- ☐ Parameter passing to sub-programs
- ☐ User definable date format

SCREEN-FORM LANGUAGE

- ☐ Programmer defined options and logic
- ☐ Multiple files open in one program
- ☐ Default or programmer processing of exception conditions
- ☐ Powerful verbs for input, display and file access
- ☐ Simultaneous display of multiple records
- ☐ Facility to call sub-programs and operating system commands
- ☐ Conditional statements
- ☐ Subroutines
- ☐ Independent of terminal type

**Sculptor for 68020
OS-9 & UniFLEX
\$995**



MUSTANG-020 Users - Ask For Your Special Discount!

MUSTANG-020

*\$1,990 \$398 \$795

PC/XT/AT/MSDOS \$695 \$139 \$299

MUSTANG-08

*\$1,295 \$259 \$495

Call or write for prices on the following systems.

XENIX SYS III & V, MS-NET, UNIX SYS III & V, ATARI OS-9, 68K, UNOS, ULTRIX/VMS (VAX, REGAJ), STRIDE, ALTOS, APRICORT, ARETE, ARM-STRONG, BLUEDALE, CHARLES RIVERS, GMX, CONVERG, TECH, DEC, CIFER, EQUINOX, GOULD III, HONEYWELL, IBM, INTEL, MEGADATA, MOTROLA, NCR, NIXDORF, N-STAR, OLIVETTI/AT&T, ICL, PERKINS ELMER, PHILLIPS, PIXEL, PLESSEY, FLEXUS, POSITRON, PRIME, SEQUENT, SIEMENS, SWTPC, SYSTIME, TANDY, TORCH, UNISYS, ZYLOG, ETC.

*** For SPECIAL LOW SCULPTOR prices especially for 6803/68XXX OS-9 Systems - See Special Ad this issue. Remember, "When they are gone the price goes back up as above!"**

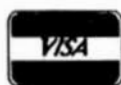
... Sculptor Will Run On Over 100 Other Types of Machines ...

... Call for Pricing ...

!!! Please Specify Your Make of Computer and Operating System !!!

- Full Development Package
- Run Time Only
- C Key File Library

Availability Legends
O = OS-9, S = SK'DOS
F = FLEX, U = UniFLEX
CO = Color Computer OS-9
COF = Color Computer FLEX



South East Media
5900 Cassandra Smith Rd. · Hickory, TN 37343
Telephone: (615) 842-4600 Telex: 5106006630



•• Shipping ••
Add 2% U.S.A. (min. \$2.50)
Foreign Surface Add 5%
Foreign Airmail Add 10%
Or C.O.D. Shipping Only

*OS-9 is a Trademark of Microware and Motorola. *FLEX and UniFLEX are Trademarks of Technical Systems Consultants. *SK'DOS is a Trademark of Star-K Software Systems Corp.

C

The C Programmers Reference Source. Always Right On Target!

C User Notes

A Tutorial Series

By: Dr. E. M. 'Bud' Pass
1454 Latta Lane N.W.
Conyers, GA 30207
404 483-1717/4570
Computer Systems Consultants

INTRODUCTION

This chapter presents an enhanced version of the `getopt` function, which is used to scan the command line. This function was originally placed in the public domain by someone unknown, but was later enhanced by Keith Bosic and others, and subsequently extended and provided by

Lloyd Zusman
Master Byte Software
Los Gatos, California

who also provided much of the text below.

COMMAND-LINE OPTION SCAN FUNCTION

Lloyd calls the enhanced version of `getopt()` by the name `egetopt()`. The default behavior of this routine is the same as that of `getopt()`, but it has some optional features that make it more useful.

These options are controlled by the settings of some global variables. By not setting any of these extra global variables, they have the same functionality as `getopt()`.

The `egetopt()` function acts like `getopt()` with the following enhancements:

The '?' which gets returned when there is an unrecognized option is now stored in a global integer called 'optbad', and the caller can set this value to anything. The initial value in 'optbad' is '?', which means that the default behavior is just like that of `getopt()`.

For example, if `egetopt()` is to return '~' instead of '?' when it encounters an invalid option, the following lines should be executed before `egetopt()` is called:

```
extern int optbad;  
optbad = (int) '~';
```

Options can begin with characters other than just '-'. There is now a global character pointer called 'optstart'. It points to a string which consists of a list of characters which can be used to begin options. The initial string that 'optstart' points to is "-", so the default behavior is like that of `getopt()`.

For example, to allow both '+' and '-' as option delimiters, put the following lines in the code before `egetopt()` gets called:

```
extern char *optstart;  
optstart = "+-";
```

Now that there's a choice of the characters that can precede options it's desirable to let the caller know what character begins a given option. In `egetopt()`, the global integer 'optchar' will now contain the character that begins a given option, or 0 if there was an error. Insert the following declaration line and check the value of 'optchar' after each call to `egetopt()`:

```
extern int optchar;  
The original getopt() writes error messages to file descriptor 2 (or to stderr, depending on the implementation). In  egetopt(), this file descriptor may point to any desired file. The global integer 'opterrfd' contains the file descriptor to use for writing error messages; it is initialized to 2.
```

As an example, to send `egetopt()` errors to go to the file " egetopt.errs", use code similar to the following before calling `egetopt()`:

```
extern int opterrfd;  
FILE *eout = fopen(" egetopt.errs", "w");  
  
if (!eout)  
{  
    /* error condition/  
  
    exit(1);  
}  
  
opterrfd = fileno(eout);
```


Some implementations of `getopt()` allow the setting of the global integer `'opterr'` to control whether error output is printed: it is initialized to 1, which enables error output (as does any non-zero value); setting it to 0 disables error output. In `eg-
getopt()`, `'opterr'` is treated the same way.

The old `getopt()` forces you to use `':'` in the string of option letters to show that a given option takes an argument. There is now a global integer called `'optneed'` which contains this value, it may be changed; `'optneed'` is initialized to `':'`.

In addition, `getopt()` is unable to handle optional option arguments. For example, if an option called `'d'` was specified as taking an argument to the program `'foo'`, `getopt()` would return the following results when invoking `'foo'` in different ways:

```
1)      foo -dABC -x ...
getopt() return:      'd'
optarg:                "ABC"
2)      foo -dABC -x ...
getopt() return:      'd'
optarg:                "ABC"
3)      foo -d -x ...
A)      getopt() return:      'd'

optarg:                "-x"
B)      getopt() return:      'd'

optarg:                NULL
```

In the case of number 3, sometimes one would prefer to get the latter results. This would allow `"-x"` to be handled as another option in the next call. In the old `getopt()`, the 3B behavior may be gotten by testing the first character of `'optarg'` and decrementing `'optind'` if this character is `'-'`. The new routine may have either behavior directly.

Since this behavior isn't always desired, `eggetopt()` checks a global integer called `'optmaybe'` which allows the programmer to control whether an option with an argument will get treated as number 3A or as number 3B above. It is used similarly to `'optneed'`. It is initialized to 0, meaning that behavior 3B is impossible by default.

The following example shows how `'optneed'` and `'optmaybe'` can be used:

```
extern int optneed;
extern int optmaybe;

optneed = (int) '!'; /* use '!' instead of ':' */
optmaybe = (int) '%'; /* use '%' for optional args */
*/
```

```
while ((c = eggetopt(argc, argv, "abc!d%x")) != EOF)
...
```

In this example, options `'a'`, `'b'`, and `'x'` take no arguments, option `'c'` takes a mandatory argument, and option `'d'` takes a non-mandatory argument. If this is contained in program `'foo'`, the following behavior will be observed when it is run:

```
foo -a -cABC -dXYZ -d -x -c -b ...
eggetopt() return:      'a'
optarg:                NULL

eggetopt() return:      'c'
optarg:                "ABC"

eggetopt() return:      'd'
optarg:                "XYZ"
>>>>>>>>>      eggetopt() return:      'd'
>>>>>>>>>      optarg:                NULL
>> NOTE >>
>>>>>>>>>      eggetopt() return:      'x'
>>>>>>>>>      optarg:                NULL
eggetopt() return:      'c'
optarg:                "-b"
```

Remember that `'optneed'` is initialized to `':'` and `'optmaybe'` is initialized to 0. This causes behavior identical to that of `getopt()` unless specifically overridden. Since the default behavior of `eggetopt()` is the same as that of `getopt()`, there is no reason why it cannot be renamed to `getopt()` and used in place of the original. It was given a new name so as not to conflict with the name of the original `getopt()` function.

Following is the source code for `eggetopt()`.

```
/*
 * eggetopt.c - Extended 'getopt'.
 *
 * Original version:
 *
 * Keith Bostic
 *
 * Current version:
 *
 * Lloyd Zusman
 * Master Byte Software
 * Los Gatos, California
 */

#ifndef EOF
#define EOF (-1)
#endif /* ! EOF */

#ifndef NULL
#define NULL (char *)0
#endif /* ! NULL */

/*
```

```

* None of these constants are referenced in the executable portion of
* the code ... their sole purpose is to initialize global variables.
*/

```

```

#define BADCH      (int)'?'
#define EMMSG      ""
#define ERRFD      2
#define MAYBESEP   (int)'\0'
#define NEEDSEP    (int)':'
#define START      "-"

```

```

/*
* Here are all the pertinent global variables.
*/

```

```

char *optarg;          /* argument associated with option */
char *optstart = START; /* list of characters that start options */
int optbad = BADCH;    /* character returned on error */
int optchar = 0;       /* character that begins returned option */
int opterr = 1;        /* if true, output error message */
int opterrfd = ERRFD;  /* file descriptor for error text */
int optind = 1;        /* index into parent argv vector */
int optmaybe = MAYBESEP; /* flag for optional argument */
int optneed = NEEDSEP; /* flag for mandatory argument */
int optopt;           /* character checked for validity */

```

```

/*
* Conditionally print out an error message and return.
*/

```

```

#define TELL(S)      { \
    if (opterr && opterrfd >= 0) \
    { \
        char option = optopt; \
        write(opterrfd, *nargv, strlen(*nargv)); \
        write(opterrfd, (S), strlen(S)); \
        write(opterrfd, &option, 1); \
        write(opterrfd, "\n", 1); \
    } \
    return (optbad); \
}

```

```

/*
* _sindex works similarly to index() and strchr().
*/

```

```

static char *
_sindex(string, ch)
char *string;
int ch;
{
    if (string)
    {
        for ( ; *string; ++string)
        {
            if (*string == (char)ch)
            {
                return (string);
            }
        }
    }
}

```

```

}
/*
* Extended getopt function
*/
int
egetopt (nargc, nargv, ostr)

```



```

int nargc;
char **nargv;
char *ostr;
{
    static char *place = EMSG;    /* option letter processing */
    register char *oli;           /* option letter list index */
    register char *osi = NULL;    /* option start list index */

    if (!nargv)
        return (EOF);

    if (nargc <= optind || !nargv[optind])
        return (EOF);

    if (!place)
        place = EMSG;

    /* Update scanning pointer. */
    if (!*place)
    {
        place = nargv[optind];
        if (!place)
            return (EOF);
        if (osi = _sindex(optstart, *place))
            optchar = (int)*osi;
        if (optind >= nargc || !osi || !*++place)
            return (EOF);

        /*
         * Two adjacent, identical flag characters were found.
         * This takes care of "m", for example.
         */
        if (*place == place[-1])
        {
            ++optind;
            return (EOF);
        }
    }

    /*
     * If the option is a separator or the option isn't in the list,
     * we've got an error.
     */
    optopt = (int)*place++;
    oli = _sindex(ostr, optopt);
    if (optopt == optneed || optopt == optmaybe || !oli)
    {
        /*
         * If we're at the end of the current argument, bump the
         * argument index.
         */
        if (!*place)
            ++optind;
        TELL("illegal option - %c");    /* byebye */
    }

    /*
     * If there is no argument indicator, then we don't even try to
     * return an argument.
     */
    if (!*++oli || (*oli != optneed && *oli != optmaybe))
    {
        /*

```

```

    * If we're at the end of the current argument, bump the
    * argument index.
    */
    if (!*place)
        ++optind;
    optarg = NULL;
}
/*
 * If we're here, there's an argument indicator. It's handled
 * differently depending on whether it's a mandatory or an
 * optional argument.
 */
else
(
    /*
     * If there's no white space, use the rest of the
     * string as the argument. In this case, it doesn't
     * matter if the argument is mandatory or optional.
     */
    if (*place)
        optarg = place;
    /*
     * If we're here, there's whitespace after the option.
     *
     * Is it a mandatory argument? If so, return the
     * next command-line argument if there is one.
     */
    else
    if (*oli == optneed)
    {
        /*
         * If we're at the end of the argument list, there
         * isn't an argument and hence we have an error.
         * Otherwise, make 'optarg' point to the argument.
         */
        if (nargc <= ++optind)
        {
            place = EMSG;
            TELL(" option requires an argument - ");
        }
        else
            optarg = nargv[optind];
    }
    /*
     * If we're here it must have been an optional argument.
     */
    else
    {
        if (nargc <= ++optind)
        {
            place = EMSG;
            optarg = NULL;
        }
        else
        {
            optarg = nargv[optind];
            if (!optarg)
                place = EMSG;
            /*
             * If the next item begins with a flag
             * character, we treat it like a new
             * argument. This is accomplished by
             * decrementing 'optind' and returning

```

```

        * a null argument.
        */
    else
    if (_sindex(optstart, *optarg))
    {
        --optind;
        optarg = NULL;
    }
    }
    place = EMSG;
    ++optind;
}
/*
 * Return option letter.
 */
return (optopt);
}

```

EXAMPLE C PROGRAM

Following is this month's example C program; it is a test driver for the command-line processing function `getopt()` just described.

```

#include <stdio.h>

/*
 * This is a program for demonstrating
 * the capabilities of getopt().
 * Run it with various combinations of
 * options and arguments on the command
 * line to see how getopt() works.
 */

#define OPT_STRING    "abc~d-e?f?"
/* Meaning:
 *
 * -a and -b take no arguments.
 * -c and -d take mandatory arguments
 * -e and -f take optional arguments
 */

#define OPT_CHARS     "+-="
/* Meaning:
 *
 * Options can begin with '-', '+', or '='.
 */

/*
 * New global variables used in getopt() only:
 */
/* string which contains valid option start chars */
extern char *optstart;
/* what getopt() returns for a bad option */
extern int optbad;
/* character which begins a given argument */
extern int optchar;
/* where getopt() error messages go */
extern int opterrfd;
/* character used for optional arguments */
extern int optmaybe;
/* character used for mandatory arguments */
extern int optneed;

```

```

/*
 * Global variables which exist in getopt() and egetopt():
 */
/* the argument of the option */
extern char *optarg;
/* set to 0 to suppress egetopt's error messages */
extern int opterr;
/* index of current argv[] */
extern int optind;
/* the actual option pointed to */
extern int optopt;

main(argc, argv)
int argc;
char **argv;
{
    int ch;

    /* errors to stdout */
    opterrfd = fileno(stdout);
    /* set this to 1 to get egetopt's error msgs */
    opterr = 0;
    /* return '!' instead of '?' on error */
    optbad = '!';
    /* mandatory arg identifier (in OPT_STRING) */
    optneed = '~';
    /* optional arg identifier (in OPT_STRING) */
    optmaybe = '?';
    /* characters that can start options */
    optstart = OPT_CHARS;

    while ((ch = egetopt(argc, argv, OPT_STRING)) != EOF)
    {
        printf("\n\toption index (optind) after egetopt(): %5d\n", optind);
        printf("\t\t\tegetopt() return value:           %c (%d)\n", ch, ch);
        printf("\t\t\tchar that begins option (optchar): %c\n", optchar);
        printf("\t\t\tactual char looked at (optopt):      %c\n", optopt);
        printf("\t\t\toption argument:                %s\n",
            !optarg ? "(null)" : optarg);
    }

    for ( ; optind < argc; ++optind)
    {
        printf("\n\targument index                %5d\n", optind);
        printf("\t\t\targument:                %s\n",
            !argv[optind] ? "(null)" : argv[optind]);
    }

    exit(0);
}
EOF

```

FOR THOSE WHO NEED TO KNOW

**68 MICRO
JOURNAL™**

SOFTWARE

USER

NOTES

A Tutorial Series

By : Ronald W Anderson
3540 Sturbridge Court
Ann Arbor, MI 48105

From Basic Assembler to HLL's

RBASIC

Since writing the last column, I have spent some time looking at RBASIC. I've found a few minor bugs and sent the report to Bob Jones. Bob has written me a letter and sent a couple of revisions. I've been working on coding some scientific functions that would be accurate to 19 digits, and I've sent the first few off to Bob. Much of the information I sent him has crossed his first letter in the mail. Unfortunately the mail from Michigan to British Columbia Canada takes about a week. Bob reported that he is interested in increasing the accuracy of the scientific functions to match the math package, and we will most likely collaborate on some improvements.

I continue to be pleased with RBASIC. I did some comparison tests with TSC Extended BASIC running on a 6809 (2 MHz). The PT68K-2 68000 with RBASIC runs on the average three to four times faster than the 6809 system doing rather math intensive calculations.

I gathered from Bob's letter that he hadn't expected much of a response from me, and that he was glad for some feedback. I mentioned having started on his logic series and he said that I was only the second person who had responded in any way. Come on, readers. If you like something in '68' Micro Journal, let

the author hear from you. It is discouraging to work at something month after month and only see one or two responses. Write Micro Journal or the author of the material that you would like to see continue, and it will have a better chance of doing so.

Having looked over the material from Bob more thoroughly, I will say that there are a few things in RBASIC that are not completely compatible with XBASIC, but the differences are mostly due to the processor differences. For example the USR function can't be the same. I won't elaborate in great detail because most of the differences are in things generally not used extensively in BASIC programs. I've urged Bob to add a few new features to RBASIC that will be extensions and will not destroy compatibility with old XBASIC programs. For example, when XBASIC first was done and I looked at it, I was disappointed to find a couple of the nice features of Bob Uiterwyk's old BASIC missing. To write three data items to a file, separated by commas, for example, in the old BASIC one simply had to:

```
100 WRITE #1, A,B,C
```

TSC's BASIC was made compatible with a BASIC from DEC, and it lacks the WRITE statement completely. Instead, as you BASIC users know, you had to:

```
100 PRINT #1, A;"",B;"",C
```

That is, you have to explicitly code the printing of the commas between the data items. I can certainly see having the PRINT statement work the same way to a file as it does to a printer, but there needs to be an easier way to write data to a file. Bob indicated that he may well be interested in a number of enhancements in the future. The other thing I have always thought was missing from XBASIC is some sort of end of file test for input files.

```
90 INPUT #1, A$(N%)
100 IF EOF(1) THEN 120
105 N%=N%+1
110 GOTO 90
120 CLOSE 1
```

Rather, with XBASIC you have to trap errors and test to see if the error is #8.

```
10 ON ERROR GOTO 1000
*
90 INPUT #1, A$(N%)
100 N%=N%+1
110 GOTO 90
120 CLOSE 1
*
1000 IF ERR=8 AND ERL=90 THEN RESUME
120 ELSE ONERROR GOTO 0
```

If you had a program with a number of files that could reach end of file at various places within the program, you would have to have a number of lines or "cases" for ERR, the error number and ERL, the line on which the error occurred, with different RESUME lines, and a catch all ONERROR GOTO 0 at the end so that other errors than end of file will be reported properly when debugging the program. It has always seemed to me as though BASIC should provide such a function so the user doesn't have to build the error trap into even a simple program to read a data file and print the contents to the screen. On the other hand, I suppose we each have our little pet "It would only take a few lines of code to add..." feature, and if Bob were to add all of them RBASIC would be big and cumbersome. Bob, I still hope you will add "mine".

A few weeks have passed since I wrote the above. Bob Jones has fixed the few little bugs that I found in RBASIC and we are continuing our discussions of how to make the math more efficient. In fact, the floating point multiply routine has been improved to be about 4.25 times faster. I'll have to scrap the few benchmarks I've run so far and wait for things to settle down a little more. Let me say that this is a very complete BASIC having full PRINT USING features, sequential and random file access, etc. I'll do a full review on it in the future. If you use BASIC for quick and dirty programs, exploratory programming, or solving problems that are very complex, you will want this BASIC.

Assembler

I received a call from a reader the other night, who had a few problems that might be common to some others of you who are just starting out on the 68000 and are new to Assembly language. I will include a short listing of a program to set an Epson printer to the 132 column (16.7CPI) mode. There are several things to point out about it. The caller's first problem was that he had written and assembled a short program but when he tried to run it SK*DOS gave him an error message that said that there was no transfer address. The transfer address is the address in the program at which it is to begin to execute. The transfer address need not necessarily be the first address in the program so the system needs some way of telling from the program code on the disk where to start executing the program relative to its first load address. To give it this information, the "start executing" address of the program must have a label. In the case of listing 1, it is the label START. Now at the end of the program the END statement tells the assembler that there is no more program to code. After the word END, you simply place the starting label, START in this case. When the program is assembled and saved to the disk as a binary file, the transfer address is coded into the file from this information. Otherwise, the code is treated as a fragment, perhaps part of another program that is to be accessed from elsewhere, and it has no transfer address, and so cannot be run by SK*DOS.

It is now about two weeks since I wrote the above. I've been plagued by a lot of work at work and some chores that had to get done around the house. I recently found a bug in my DDUMP utility published here a few months ago. DDUMP, you may

remember, does a HEXADECIMAL and ASCII dump of a disk file, sector by sector. I had used it primarily to dump binary files, but one day not too long ago, I decided to use it to look for something funny in a text file. I found that it exited after the first sector. I made a few changes and then found that it insisted on repeating the first sector 5 times and then went on to the next one. After a little hair pulling I discovered that it would foul up if there happened to be a TAB at the start of the sector.

Text files in SK*DOS use a technique called space compression. Rather than put 10 or 12 \$20 (space) characters in a file, SK*DOS automatically compresses consecutive spaces by using a horizontal tab character \$09 followed by a space count. If there are two spaces together, of course, you come out even, using two characters. However, any number from 3 to 255 spaces can be compressed to two characters by this method. What I eventually figured out, is that the FSKIP function of SK*DOS doesn't perform its function if FREAD has gotten the system into the middle of a space expansion while reading the file. FSKIP is supposed to skip the remainder of the current sector so the next sector will be read on the next FREAD call. It worked fine except when in the middle of a space expansion. I decided to tell SK*DOS not to expand spaces, since I wasn't using the read function to get the data, but was taking it directly from the file control block. All that is needed is to stuff \$FF into the 59th byte of the file control block after opening the file but before reading any data. While I was looking at it I cleaned up the code a little. The listing of version 2.0 is included here. I've improved the error handling and trapped error #8 as end of file. That error occurs when you try to read

past the last valid sector of a file. Essentially I have done what I was complaining about having to do in BASIC above.

I have a few copies of the SK*DOS version of PAT, my editor out for testing. I have held off on distributing it because I wanted to check it out personally on a PT68K-2 system using a monitor and IBM keyboard. From one source I hear how slow that system is compared to a terminal, and from another I hear how fast it is. I want to check it out for myself and perhaps write a driver that will allow some fancies on the screen. My other motive for this is that Dan Farnsworth of Palm Beach Software has sent me his EDDI text editor that runs with this setup. I have ordered the monochrome board and IBM clone keyboard for my system. I have a monitor kicking around somewhere, so I will be able to run the system in that mode and free up a terminal. Actually I have still another motive, that of the possibility of doing some graphics for my work projects. That is a long term goal.

I've been using RBASIC for several days to analyze some peculiar problems at work, and have not found any further bugs in it. It is nice to have a BASIC that runs on the 68K systems for testing ideas and techniques for my work projects. Along that line, I have a most peculiar problem with combinations of weights. Essentially I can have 5 different weights but I can use only three or less weights at any one time. I can use one of the weight values repeatedly as in 7,2,2 etc. The problem is to cover the greatest range of weight in unit steps with various combinations of these weights. I found a set consisting of 1, 2, 3, 7, and 18 that would let me make combinations of three or less weights for values from 1 to 28 with no values skipped. Many of the combi-

nations are redundant, and I would guess that there would be other sets that might span a larger range. I am perplexed that I could find no logical approach to calculating the best set of weights. Of course if there were not a limitation on the number of weights a simple binary progression would do the job.

In a similar problem I have 6 weights to be taken 1 2 or 3 at a time. I came up with 1, 2, 3, 7, 11, and 26, which gets me 0 to 40 in steps of 1. Are there any mathematicians or logicians out there who can give me a clue as to how to approach this problem to get the largest coverage from a given set of weights? A set of 7 taken 3 at a time get me from 0 to 52. They are 1, 2, 3, 7, 11, 15, and 34. If any of you can figure out a good series that will go farther in continuous steps of 1 I'd like to hear about it. The matter is now academic because of other limitations on the problem, but it is still of interest.

Last Minute News

I just received a new version of RBASIC in which the floating point divide has been speeded up by more than a factor of 2, and the display of integer numbers has been cleaned up. The original had a problem if the user specified too many digits for display. It would sometimes display with a slight conversion error showing 15.9999999999999978 rather than the actual value of 16 for example. Bob has cured that problem in this latest version as well.

I just returned from a vacation trip with my wife to New Zealand with a stop in Hawaii on the return trip. We visited John Spray in Auckland and did a whirlwind tour of the country. I can say that it is truly a beautiful place. We thoroughly enjoyed our visit there. I had the opportunity to

drive a right-hand drive vehicle on the left side of the road. That wasn't as difficult as it might seem, but I did have a hard time with the manual transmission shift lever on the left and turn signal on the right. I kept trying to shift gears with the turn signal. We found a strange mixture of prices. Gasoline (or Petrol) costs about twice what it does here in the U.S. but a good dinner in a restaurant about half as much.

More to the point of this column, John and I visited several computer stores while we were there. Most of them are just like Computerland and their clones with slick equipment setups and sales people to tell you all about the advantages of their products. We did visit one that was explicitly for the "hacker". They had computer boards laid out on tables and very low prices. We found out that many of the boards were not in working condition and a purchaser could either buy a tested one at a much higher price or buy two or three in hopes of getting one that works or can be fixed. There were also quantities of used disk drives and the like for sale under the same terms. I'd say that computing there has gone pretty much down the IBM and clone road just as it has here. Outside of the industrial control area there is very little else around.

Well, the trip has taken three weeks of my time so this material is about to be late. Larry Williams called this morning to see if I had any material ready to mail in, so I guess I'll run this through the appropriate conversion program and add the two listing files to a disk so it can be mailed on time.

```

*****
* DISKFILE DUMP PROGRAM *
* *
* COMMANDS: *
* N XXXX NEXT PAGE TO BE DUMPED *
* B BACK A SECTOR *
* F FORWARD A SECTOR *
*****

```

* SK*DOS / 68K EQUATES FOR USER PROGRAMS

0000A029	GETCH	EQU	\$A029	Get input character with echo \$lts)
0000A023	GETNAM	EQU	\$A023	Get filename into FCB
0000A024	DEFEXT	EQU	\$A024	Set default extension
0000A005	FOPENR	EQU	\$A005	Open file for read
0000A001	FREAD	EQU	\$A001	Read a byte
0000A011	FSKIP	EQU	\$A011	Next Sector
0000A031	TOUPPR	EQU	\$A031	Conv't char in D5 to Upper Case
0000A02F	HEXIN	EQU	\$A02F	Input hexadecimal number
0000A03A	OUT2H	EQU	\$A03A	Output 2 hex digits
0000A03B	OUT4H	EQU	\$A03B	
0000A03C	OUT8H	EQU	\$A03C	Output 8 hex digits
0000A034	PCRLF	EQU	\$A034	Print CR/LF
0000A037	PERROR	EQU	\$A037	
0000A036	PNSTRN	EQU	\$A036	Print string (Without CR/LF)
0000A035	PSTRNG	EQU	\$A035	Print CR/LF and string
0000A033	PUTCH	EQU	\$A033	Output character
0000A000	VPOINT	EQU	\$A000	Point to SK*DOS variable area
0000A01E	WARMST	EQU	\$A01E	Warm start

*
*

000000 A000	START	DC	VPOINT	GET POINTER
000002 284E		MOVE.L	A6,A4	FCB POINTER
000004 A023		DC	GETNAM	GET FILENAME
000006 6500 0090(00098		BCS	HELP	
00000A 7801		MOVE.L	#1,D4	CODE FOR .TXT
00000C A024		DC	DEFEXT	
00000E A005		DC	FOPENR	
000010 197C 00FF 003B		MOVE.B	\$\$FF,59(A4)	SET NO SPACE COMPRESSION
000016 A001	LOOP	DC	FREAD	
000018 6618 (00032		BNE.S	ERROR	
00001A 204C		MOVE.L	A4,A0	
00001C D1FC 0000 0060		ADD.L	#96,A0	POINT AT SECTOR INFO
000022 611C (00040		BSR.S	OPAGE	
000024 A029		DC	GETCH	
000026 A031		DC	TOUPPR	
000028 0C05 0045		CMP.B	#'E',D5	
00002C 6710 (0003E		BEQ.S	EXIT	
00002E A011	CONTIN	DC	FSKIP	
000030 60E4 (00016		BRA	LOOP	
000032 0C2C 0008 0001	ERROR	CMP.B	#8,1(A4)	IS END OF FILE?
>000038 6700 0004(0003E		BEQ	EXIT	
00003C A037		DC	PERROR	
00003E A01E	EXIT	DC	WARMST	

*
* ROUTINE TO OUTPUT A PAGE IN HEX AND ASCII

*

000040 A034	OPAGE	DC	PCRLF	
000042 4240		CLR.W	D0	LINE COUNTER
000044 A034		DC	PCRLF	
* LOOP FOR LINES				
000046 323C 000F	LLOOP	MOVE.W	#15,D1	COUNTER FOR CHARACTERS
00004A 1800		MOVE.B	D0,D4	
00004C E904		ASL.B	#4,D4	ADDRESS OF FIRST BYTE OF LINE PAGE
00004E A03A		DC	OUT2H	
000050 183C 0020		MOVE.B	\$\$20,D4	
000054 A033		DC	PUTCH	SPACE
000056 A033		DC	PUTCH	SECOND SPACE
* INSIDE LOOP FOR 16 CHARACTERS IN HEX				
000058 1818	CLOOP	MOVE.B	(A0)+,D4	
00005A A03A		DC	OUT2H	OUTPUT FIRST BYTE
00005C 183C 0020		MOVE.B	\$\$20,D4	


```

000060 A033          DC      PUTCH      SPACE
000062 57C9 FFF4{00058 DBEQ    D1,CL P      CHARACTERS
      * NOW DO ASCII CHARACTERS, "." FOR NON PRINTABLE

000066 183C 0020      MOVE.B    #$20,D4
00006A A033          DC      PUTCH      EXTRA SPACE BEFORE ASCII
00006C 91FC 0000 0010 SUB.L     #16,A0
000072 323C 000F      MOVE.W    #15,D1      RELOAD COUNTER FOR CHARACTERS
000076 1818          ALOOP    MOVE.B    (A0)+,D4
000078 0244 007F      AND        #$7F,D4      MASK OFF HI ORDER BIT
00007C 0C04 0020      CMP.B     #$20,D4      IS IT PRINTABLE?
000080 6C04 {00086    BGE.S     AL1        IF YES
000082 183C 002E      MOVE.B    #'.' ,D4      ELSE PRINT PERIOD
000086 A033          AL1      DC      PUTCH
000088 57C9 FFEC{00076 DBEQ    01,ALOOP
00008C A034          DC      PCRLF
00008E 5240          ADD        #1,D0
000090 0C00 0010      CMP.B     #16,D0
000094 66B0 {00046    BNE      LL P      NEXT LINE
000096 4E75          RTS

000098 49FA 0006{000A0 HELP LEA      HLPMSG(PC),A4
00009C A035          DC      PSTRNG
00009E A01E          DC      WARMST

0000A0 5379 6E74 6178 HLPMSG DC.B     "Syntax: DDUMP FILENAME Defaults are active", $0D,$0A
0000D2 616E 6420 2E54      DC.B     "and .TXT extension.", $0D,$0A,$0A
0000E8 4444 554D 5020      DC.B     "DDUMP dumps a disk file to the terminalerror", $0D,$0A
00011C 6174 2061 2074      DC.B     "at a time. The dump displays 16 bytes in HEX followed by", $0D,$0A
000157 7468 6520 4153      DC.B     "the ASCII representation of the same 16 bytes.Non-", $0D,$0A
00018D 7072 696E 7461      DC.B     "printable characters are displayed as periods.", $0D,$0A
0001BD 4174 2074 6865      DC.B     "At the command prompt, P will cause the(forward)", $0D,$0A
0001F5 7365 6374 6F72      DC.B     "sector to be displayed, and B (back) will display", $0D,$0A
00022C 7072 6576 696F      DC.B     "previous sector. E will Exit the programto", $0D,$0A
00025F 4620 7768 656E      DC.B     "F when the last sector of the file has been", $0D,$0A,$0A

      END      START

```

0 ERRORS DETECTED

```

      * PROGRAM TO SET EPSON TO 16 CPI
      *
0000A033 PUTCH EQU $A033
0000A000 VPOINT EQU $A000
0000A01E WARMST EQU $A01E
00000CCB DEVOUT EQU 3275
00000000 ORG $0000
00000000 A000 START DC VPOINT
000002 1D7C 0002 0CCB MOVE.B #2,DEVOUT(A6)
000008 183C 000F MOVE.B #$0F,D4
00000C A033 DC PUTCH
00000E 1D7C 0000 0CCB MOVE.B #0,DEVOUT(A6)
000014 A01E DC WARMST
      END      START

```

0 ERRORS DETECTED

+++

FOR THOSE WHO NEED TO KNOW

68 MICRO
JOURNAL™

Logically Speaking

Most of you will remember Bob from his series of letters on XBASIC. If you like it or want more, let Bob or us know. We want to give you - what you want!

The Mathematical Design of Digital Control Circuits

By: R. Jones
Micronics Research Corp.
33383 Lynn Ave., Abbotsford, B.C.
Canada V2S 1E2
Copyrighted © by R. Jones & CPI

SOLUTIONS TO TEST FOURTEEN-A

Relays Operated	Code	X		Z
		0	1	
0	0	0	1	
1L	1	2	3	1
S1	2	2	-	1
2L	3	-	4	
3L	4	5	-	1
S3	5	5	-	1

X		Z
0	1	
0	1	
1	3	1
2	-	1
3	2	

6 contacts
9 springs

(a)

Combine 2,4,5

Relays Operated	Code	X		Z
		0	1	
0	0	0	1	
1L	1	2	3	1
S1	2	2	1	1
2L	3	-	4	
3L	4	5	-	1
S3	5	5	1	1

X		Z
0	1	
0	1	
1	3	1
2	1	1
3	4	
4	2	1

8 contacts
13 springs

(b)

Combine 2,5

Relays Operated	Code	X		Z
		0	1	
0	0	0	1	1
1L	1	2	3	1
S1	2	2	1	1
2L	3	-	4	
3L	4	5	-	1
S3	5	5	1	1

X		Z
0	1	
0	1	1
1	2	1
2	3	
3	0	1

6 contacts
10 springs

(c)

Combine 0,2,5 renumber 3,4 → 2,3

Relays UNOperated	Code	X		Z
		0	1	
0	0	1	0	
1L	1	2	-	
2L	2	-	3	1
S2	3	-	3	1

		X		Z
		0	1	
0		1	0	
1		2	-	
2		-	2	1

4 contacts
6 springs
(h)

Combine 2,3

Be careful here! Note that we're keeping track of the number of UNOPERATED relays.

Relays Operated	Code	X		Z
		0	1	
0	0	0	1	
1L	1	2	3	1
S1	2	2	-	1
2L	3	4	-	1
S2	4	4	-	1

		X		Z
		0	1	
0		0	1	
1		2	2	1
2		2	-	1

3 contacts
5 springs
(i)

Combine 2,3,4

This is the inverse of the specs, and should be graphically complemented to give the desired output conditions. NOTE : Line-1 needs no contacts.

Relays Operated	Code	X		Z
		0	1	
0	0	0	1	
1L	1	2	3	
S1	2	2	4	
2L	3	5	6	
S1 + 1L	4	-	7	
S2	5	5	8	
3L	6	2	6	
S1 + 2L	7	9	-	1
S2 + 1L	8	9	10	1
S1 + S2	9	9	-	1
S2 + 2L	10	-	11	
S2 + 3L	11	9	11	1

		X		Z
		0	1	
0		0	1	
1		2	3	
2		2	4	
3		5	6	
4		-	7	
5		5	8	
6		2	6	
7		7	-	
8		7	9	
9		-	10	
10		7	10	

19 contacts
29 springs
(j)

Combine 7,9
Renumber 10,11 → 9,10

NOTE : In line-6, column X=0, I've not opened up a line for S3, because this circuit is interested in sets of TWO only. Any other size is all the same to this network, so I can save myself a lot of work by making all sets larger or smaller than TWO take on the same name, namely S1, which I now choose to interpret as "One set of any size except TWO". Similarly in Line-11.

If you attempted each and every one of the above, especially the last five (which are reasonably tough), you should have a really good idea how to create iterative tables by now! They were good practice, anyway! So, if everyone's ready, let's look at a final high-powered technique for manipulating a prototype table.

Mile 18 - heading for Mile 19

MULTIPLE ASSIGNMENT

In our earlier study of iterative networks we've always produced a prototype cell with one input-line for power, which is switched from one level to another as it proceeds through the cascaded network. Eventually, if it's not cut off along the way, power arrives at one or more output-lines which are connected together.

Let's pretend we've developed a table which switches between eight different levels, numbered from 0 through 7, where each level represents a different piece of info being passed from one relay to the next along our chain of "n" relays. I don't think I'm far wrong when I say that all of you know that eight different numbers (or data-codes) CAN be transmitted in binary along only THREE wires, instead of the EIGHT original lines. This is just what we're going to attempt in our final session with iterative networks. We're going to "code" our info, using multiple line assignments, instead of our usual "one line = one piece of info". Of course we're going to this scientifically, and develop codes which won't create any conflict one with the other. And just so we can tell whether "multiple" coding of a table has produced any improvement over "single" coding, we obviously have to have some standard of comparison.

SPRINGS AND CONTACTS

This will be based on the number of "springs" and "contacts" in both implementations. You already know what a "contact" is, so I'll simply define a "spring" as a terminal on a relay-contact to which a wire can be connected. An ordinary NO-contact or NC-contact has TWO springs, because two wires can be connected to it, one at each end of the contact. A transfer-contact, on the other hand, has THREE springs, because one wire can be fastened to the common centre-point, and one each to the NO and NC contacts between which it can be switched.

So without further ado, let's get on with a new set of specifications with a slightly different twist to vary our diet a little, and compare the two methods of assignment. Here then is

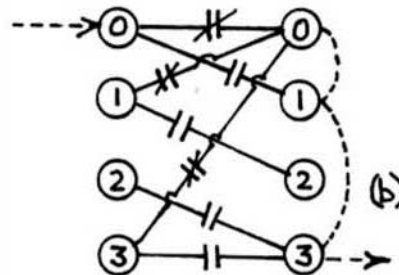
ITERATIVE NETWORKS - MULTIPLE ASSIGNMENT - EXAMPLE 1

We're called on to design the prototype cell for a network of "n" relays whose output-line is normally powered-up, but which has to shut off iff exactly ONE set of TWO relays becomes operated.

This time I don't think you'll need a step-by-step explanation of the table shown in Diagram 96a, especially if you take time out to actually create it yourself. Just a brief pointer to the Z-column, where you'll notice that we're calling for power in all lines except where the set of 2 relays is formed. The normal table is necessary as the first step in our new procedure, and I've already drawn the prototype cell in 96b for comparison purposes later.

Relays Operated	Code	X			Z
		0	1	2	
0	0	0	1	1	
1L	1	0	2	1	
2L	2	-	3		
3L or more	3	0	3	1	

(a)



(b)

Diagram 96

If we look at the network, we see that X has a total of seven contacts, which, when grouped to form transfer-contacts, require eleven springs to form the circuit - 3 transfers of 3 springs each, and 1 NO-contact of 2 springs. With me so far? Good! Note that our prototype table shows that an output is required for lines, or states, 0, 1 and 3 (which is why outputs 0, 1 and 3 are joined together in 96b), and it's at this point that we begin our multiple assignment coding.

CREATING A MULTIPLE-ASSIGNMENT PROTOTYPE TABLE

This begins with Diagram 97a, where our original "Code" column has now been moved out to the extreme left. Keep in mind that Z calls for an output in lines 0, 1 and 3 - highest number therefore being "3" - so everywhere that 0, 1 or 3 appears in our original table (in the "code" column or either of the X-columns) we're going to write "3" (the highest number called by Z).

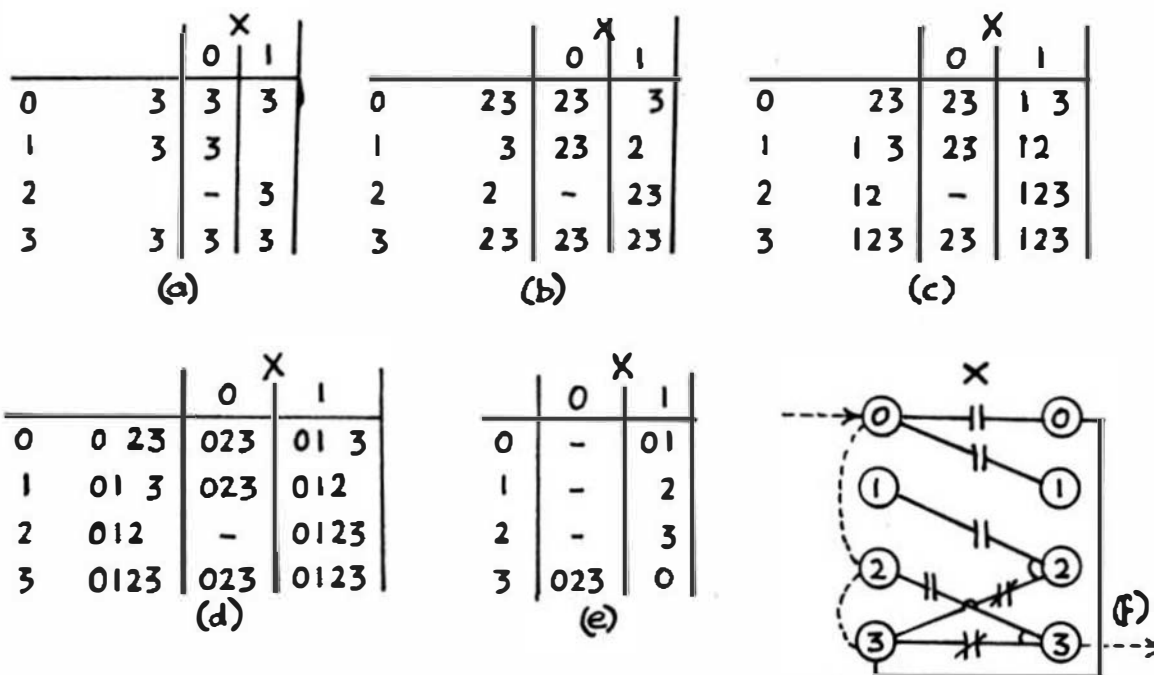


Diagram 97

The procedure we're going to follow isn't difficult, but we MUST be careful not to make any mistake as we go along. It's a continuous repeat of a very simple step, so stay close and it'll all come out OK. Still with 97a then, we've just set out a new coding, both in the "Code" column and in both internal columns, and at the moment it seems that we have 3s everywhere. Now, treading very warily, let's look at column X=0 and ask ourselves "For every 3 in this column, is there a consistent digit in the code-column?" We see that this IS so, namely that for every 3 in this column there's a 3 externally in the code-column. OK then, let's move to the X=1 column and ask the same question. This time we have to answer "No", because, although the 3s in lines 0 and 3 both line up with an external 3, the 3 in line-2 doesn't even HAVE an external number, let alone one which is the same for ALL the 3s in this column.

ADDING CODE-2

We'll rectify this situation at once, by inserting IN THE CODE-COLUMN the next lowest number (ie, 2) directly opposite the 3s in column X=1 (see Diagram 97b). So now we DO have a consistent number which lines up with ALL our 3s - namely, an external 3 lining up with the 3s in column X=0, and an external 2 lining up with the 3s in column X=1. Note that we keep these external numbers in neat columns! Now the 0 at the far left corresponds to code-23 in the code-column, so everywhere there's a 0 in the original internal columns, we MUST ensure that it gets changed to 23. Similarly, far-left-1's code is unchanged at code-3, so we leave this untouched in 97b's internal columns. Far-left-2 corresponds to code-2 now, so original internal 2s, of which there's only one, gets recorded as 2 in 97b, and finally, far-left-3 corresponds to code-23, so all original internal 3s get changed to 23. This completes the compilation of table 97b.

Having just added 2s all over the place, we must now concentrate on this figure. As before, beginning with column X=0, we ask our standard question, only this time we'll ask it about the 2s in 97b. And what do you know? The 2s in the X=0 column DO have a consistent external figure lining up with them, namely the 3s in the code column. Ho hum! Let's move to column X=1 instead, and repeat our question. The answer has to be "No", because the first and last 2s in this column line up with an external 3, while the lower two 2s line up with an external 2. No consistent line-up at all!!

NOW FOR CODE-1

Nothing for it but to FORCE an alignment, and thereby create 97c, by entering 1s in the code column to line up with the 2s in column X=1. This means that we have to amend our INTERNAL codings too! Only codes 1, 2 and 3 got changed this time, so let's begin with far-left 1 and corresponding code-13, which tells us to change our original internal 1s to 13. Similarly far-left-2 causes original 2s to be replaced with 12, and far-left-3 causes original 3s to become 123. And that's 97c done!

AND FINALLY CODE-0

Now to process the internal 1s. Column X=0 doesn't have any, so there's no problem there, but column X=1 has 1s running all the way from top to bottom. Nowhere is there a SINGLE, consistent, external figure in the code-column to line up with these four 1s, so let's FORCE it once more by putting the next lowest figure (ie, 0) in the code-column (see 97d), and just as before we'll update the table internally. For example, far-left-0, code-023, tells us to replace all original internal 0s with the code 023, and so on for the rest of the table. Final check of 97d coming up! "In column X=0, do our 0s line up consistently?" YES, they do, with the 3s in the code-column. A check of column X=1, of course, shows internal 0s lining up consistently with external 0s. So we've come to the end of the process, having now successfully coded our original table into a multiple assignment form.

Don't relax just yet though! There's more to come before we can draw our multiple-assignment prototype!! We must first convert 97d to what's known as a "terminal connection table", the procedure again being simple and straightforward, thank goodness! See 97e to follow my explanation.

THE TERMINAL CONNECTION TABLE

Here we'll begin with the skeleton of a prototype-table, with just the far-left codes of 97d in the code-column. Next, we're going to be looking from the viewpoint of the code-column of 97d INWARDS to its X-table. Let me elaborate! We'll start with the 0s in 97d's code-column, and ask "What figure(s) in 97d's X=0 column does 0 consistently line up with?" As there are four external 0s, there is NOTHING in column X=0 that lines up with all four, so we enter a "-" in 97e in column X=0 in line with the 0 in ITS code-column. Back to 97d, to ask ourselves "And what does an external-0 line up with consistently in the X=1 column?" Here it lines up with 01, so in 97e in the X=1 column we write 01 to line up with external-0.

Now for the 1s in 97d's code-column. They line up with nothing in X=0, so we enter a "-" in 97e's X=0 column opposite code-1. These external-1s line up CONSISTENTLY only with the 2s in column X=1, so this is what we record in 97e's X=1 column opposite code-1. Similarly, we record the fact that 97d's external-2 lines up with nothing in column X=0, but with 3 in column X=1.

LET'S THINK A LITTLE ABOUT SOME OF THE CODES

Finally we record the fact that 97d's external code-3 lines up consistently with 023 in column X=0, but, BECAUSE WE HAVE AN ENTRY IN COLUMN X=0, in the case of the X=1 column we'll do a little bit of thinking first. First let's observe that although 97d's external-3 lines up consistently with internal-01, the reverse cannot be said to be true, namely, internal-23 doesn't consistently line up with external-3. Under these circumstances, where a line-up only half-complies, we are free to enter 0, 1, 01 or nothing at all in 97e's last remaining slot. Which shall we enter to give us the best advantage? Now's the time to remember an earlier rule about prototypes, which said that if the same entry appears in BOTH X-columns then no contacts are necessary, a straight wire-connection being sufficient. With this rule in mind, our choice is obvious! We'll insert a 0 in this location, to go with the 0 in the X=0 column. 1 is no use to us, as it would only mean an extra contact.

Sometimes we can complete our connection table a little more quickly by filling in all the "-"s first, if there ARE any in the multiple-assignment table developed. For instance, in 97d, there's a "-" in column X=0 opposite code-012, so we could right away (in our connection table) put a "-" in rows 0, 1 and 2 in column X=0, which would leave only one slot to be completed in this column, plus, of course, we'd have to do column X=1. No "-"s in column X=1, so we can't take a short cut there. This method could save some time, and possibly a lot of checking of external codes against internal ones.

DRAWING THE PROTOTYPE CELL

With the terminal connection table all complete, we're ready to draw the prototype at last! This is shown in 97f, the interpretation being the same as before, namely, input-line 0 goes nowhere via a NC-contact, but goes to both terminals 0 and 1 via NO-contacts. And so on for the rest of the table, noting that input-line 3 goes to output-lines 2 and 3 via NC-contacts, but to output-line-0 via a direct connection (as 0 appears in both columns). As far as power input is concerned ... remember it always came in at input-line-0? Our multiple-assignment table in 97d shows that our old code-0 now corresponds to code-023, so we connect power to input-lines 0, 2 and 3 of our first cell. Power-output is ALWAYS taken off at the highest-numbered output-line only, in our case output-line-3.

Observe that our new prototype cell only requires six contacts and ten springs, compared to the original seven contacts and eleven springs. In view of all the work we've just done, you may feel that this isn't much of a saving, but always remember ... this is how much we save PER CELL, so if we had 199 relays in the chain we'd have saved 199 contacts altogether, PLUS the necessary wiring!! Sometimes the work doesn't save anything at all, and we just end up with a DIFFERENT circuit performing the same function as before. Only one thing is reasonably certain - we won't require MORE contacts or springs!

Here's an example which shows how big the saving in contacts can really be!

ITERATIVE NETWORKS - MULTIPLE ASSIGNMENT - EXAMPLE 2

The specs call for a prototype cell to be designed for a circuit of "n" relays which will give a continuous output, UNLESS two consecutive sets are separated by more than one UNOPERATED relay, in which case the power gets cut off.

You're so good at this stuff now, that I don't think a detailed description will be necessary, and you should be able to feel your way through the problem with the aid of the tables of Diagram 98 alone. I will, however, get you started on the first multiple-assignment table of 98c. This is compiled by writing our original codes to the far-left, then, noting in 98a that Z has an entry for all codes 0, 1, 2 and 3, we enter the highest of these numbers (ie, 3) in all locations in 97c where 97a has 0, 1, 2 or 3, whether in the code-column or in the internal table.

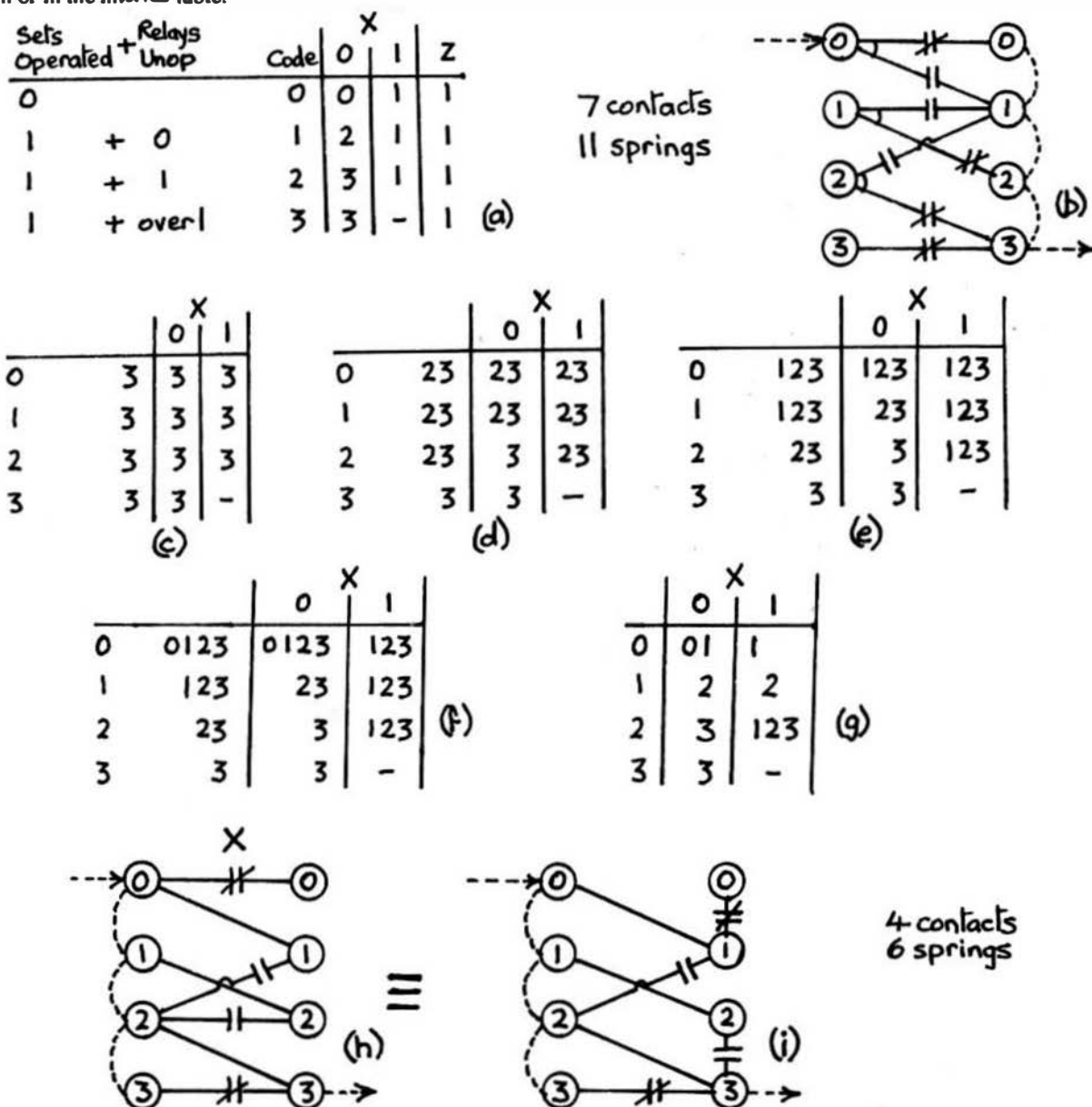


Diagram 98

On second thoughts, maybe I WILL talk you through the construction of Diagram 98g from 98f. First, we'll tackle the "-"s by noting that we have one in 98f, unluckily for us only opposite code-3, so in 98g's code-3, column X=1 we insert a "-". OK, now to 98f's code-column, commencing with 0. This lines up with 01 in the X=0 column, so this is what we record for 98g's code-0 in column X=0. Now, in 97f's X=1 column, 0 lines up with 123, BUT THE REVERSE ISN'T TRUE, as 123 lines up with other rows as well. This means we can pick and choose here, so we'll select only the 1 of 123, to change the 1 already in the X=0 column into a simple wire-connection.

Looking at 98f's external-1 we note that it DOES consistently line up with 2 in column X=0, so let's record this in 98g. Again we have a situation similar to that for our 0-row, namely, code-1 lines up with 123 in column X=1, but not in reverse, so we're free to pick and choose here too. So let's select the 2 of 123 for 98g's column X=1, to go with the 2 in column X=0.

Coming now to 98f's code-2, we note that it, too, only half-complies. That is, although it lines up with 3 in column X=0, internal-3 doesn't consistently line-up with external-2, so let's withhold judgment for a moment while we look at column X=1. No doubt at all here, 2 lines up with 123, which we record in 98g's column X=1. And so, of course, we have no option but to put our 3 in column X=0 in order to neutralise the 3 in column X=1. OK?

Finally, in 98f, external-3s line up with internal-3s in column X=0, and we end up by recording this fact.

When we come to power up our prototype in 98h, we observe that, according to 98f, our original code-0 is now code-0123, so we connect power to input-lines 0, 1, 2 and 3 of our first prototype cell in the chain, and connect our output-device to output-line 3 of the final cell, this being the highest code-number.

RE-ARRANGING THE CONTACTS IN A PROTOTYPE CELL

Observe how I've shuffled around the contacts of our prototype to form a new cell in 98i. Because input-line-0 is connected DIRECTLY to output-line 1, it doesn't matter in theory whether one end of the upper NC-contact is connected to input-line-0 OR to output-line-1, AS THEY'RE THE SAME POINT ELECTRICALLY. In actual practice though it makes a tremendous difference, as I can now use a single transfer-contact with its centre-point connected to output-line-1. Similarly, I can transfer one end of the NO-contact on input-line-2 so it connects instead to output-line-3, to produce another transfer-contact.

Alternatively, commencing once more with 98h, I could equally as well have shifted the end of the NO-contact from output-line-1 to input-line-0, to form a transfer with its centre on input-line-0. And the end of the NC-contact presently connected to output-line-3 COULD have been shifted to input-line-2. Or any combination of all these shufflings around!

Now for the big surprise! If we count the number of contacts and springs in our new prototype, we find that we've reduced the original 7-contacts/11-springs to a VERY compact 4-contacts/6-springs!! A substantial saving indeed! Especially when you consider that we can now use smaller-sized relays. And only six wires going to each relay instead of the original eleven!! So you see, it IS worthwhile to try multiple-assignment if you're into iterative networks at all.

In fact, it's such an interesting technique, that I'd now like you to do

TEST FOURTEEN-B

Commencing with the regular prototype tables of TEST FOURTEEN-A, convert them to multiple-assignment tables, and compare the original contact/spring rating with the new.

CHIT-CHAT TIME AGAIN

And that, I'm afraid, is just about all I can tell you about iterative networks. This stage of our journey is a fairly critical one, as the next few miles are likely to be a little rough, though I'll TRY to explain things as simply as I can. If you can push your way through that stretch of jungle, you'll not only have a DEEP understanding of some of the most powerful aspects of network design, but will know a LOT better what Boolean Algebra IS and what it ISN'T. However, just in case it all proves a little too much for the lesser-experienced of you, I've arranged for helicopters to lift you over this stretch, and deposit you in a jungle holiday-camp where you can swat mosquitoes, relax, swat mosquitoes, bask in the sunshine, swat mosquitoes, etc., until I and your more fool-hardy companions catch up to you again. In other words, you can skip the next few sections, dealing with Boolean matrices (of all things), if you wish! I'd recommend though that you come with us for a little way at least!

... End of Mile 18. At Mile-19 marker, wondering just what sort of creepy-crawlies and other unmentionables lie waiting in the jungle ahead. Shudder! Shudder! Not that I'd want you to lose any sleep over it. Remember the helicopters!

+++

FOR THOSE WHO NEED TO KNOW

**68 MICRO
JOURNAL™**

INTERFACING TO THE MOTOROLA 88000 RISC CHIP SET



BY: Terry Lawell & Sang Quan
MOTOROLA Inc.
3501 Ed Bluestein Blvd.
PO Box 6000, Austin, Tx. 78762

MOTOROLA INC.

With the recent introduction of the Motorola 88000 family of high performance RISC architected ICs, a new era in performance oriented microprocessing has begun (refer to figure 1). Like many other performance oriented systems though, the performance of the 88000 family is determined by many factors. These include factors such as the raw processing power of the cpu, the cpu architecture (Von Neumann or Harvard), memory management, code/data caching and the interfacing techniques to the user environment. It seems that in many systems designs interfacing to the user environment does not receive the attention it deserves. In developing the architecture for the 88000 family, Motorola paid great attention to this particular aspect of the system design and developed a new bus interface called the M-Bus. Prior to discussing the details and attributes of the M-Bus, a brief discussion of the 88000 family and philosophy behind the architecture will give the reader a better appreciation for the approach used in the architecture of the M-Bus.

The 88000 family of RISC devices includes the 88100 CPU which provides the integer and floating point processing power for the family and two 88200 CMMUs which provide the memory management and data caching for the data and instruction code. The 88100 uses a Harvard Architecture (totally separate paths for the instruction code and the data) for its external interface and is intended to interface directly to the CMMUs via the P-Bus interface. This interface is optimized for connection to the CMMUs to insure zero wait state access of code and data, but does not preclude the user from connecting the 88100 directly to other memory or I/O devices. When connected to the CMMUs, the 88100 can support up to 4 CMMUs on the code bus and 4 CMMUs on the data bus. The internal architecture of the 88100 is architected for pure performance (refer to figure 2). The internal

architecture is organized into four independent execution units with their own separate pipelining capabilities. The 88100 sports a completely synchronous bus interface with separate 32 bit data and address lines and operates at 20mhz. The cpu is capable of executing one instruction (including floating point) every clock cycle while concurrently loading or storing data at a rate of up to one word per clock cycle. With the configuration shown in figure 2 Motorola can guarantee the customer that as cpu clock speeds are scaled upwards, the CMMU speeds will be scaled to match. The 88100 supports many more features than just those mentioned here, but the ones mentioned will give the reader at least a glimpse of what the 88100 is capable of providing.

The 88200 CMMU provides the memory management and caching functions for both the data path and the code path. As mentioned earlier, a typical system would include two CMMUs, one on the data side and one on the code side of the cpu. The organization of the

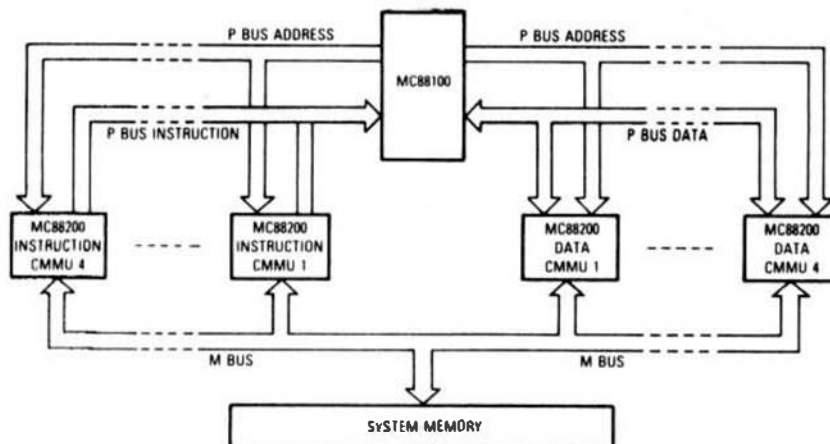
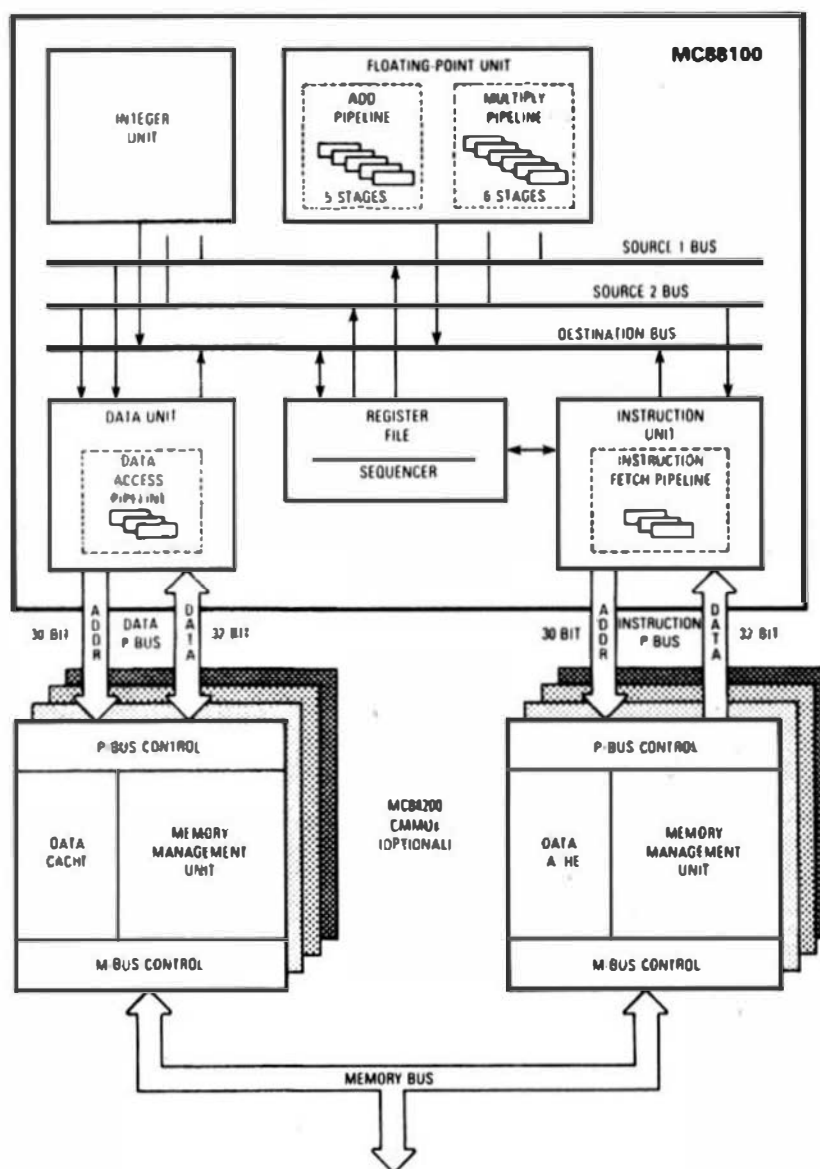


Figure 1 System Block Diagram

Figure 2 88100/88200 Block Diagram



CMMU is quite modular and divided cleanly between the memory management and data caching functions (refer to figure 2). The memory management portion of the CMMU consists of two address translation caches which provide the user with different levels of granularity for address hit detection. The PATC (page address translation cache) provides the user with 56 entries each covering 4k bytes of address space and is updated automatically on an ATC miss

by table walks. The BATC (block address translation cache) has 10 entries that provide 512k bytes of address translation each and are loaded and updated by software. The data cache portion of the CMMU, though a totally separate unit from the memory management unit (MMU), works in parallel with the MMU to provide a "no wait state" data caching function. The data cache is organized as a 16-kilobyte four way set associative physical cache. To achieve

the "no wait state" capability, the selection of the data cache set is performed at the same time the MMU is doing the logical to physical address translation.

In most environments, the CMMU acts as the user's interface to the 88100 cpu via the M-Bus interface. The remainder of this article will discuss the details of the 88200 CMMU and the user interface M-Bus.

CMMU Description

To effectively play the role of an active buffer between the processor and the memory subsystem, the CMMU implements interfaces to the P-Bus on the processor side and M-Bus on the memory side. Logical addresses issued by the processor are presented to the CMMU on the P-Bus, translated (if valid) into physical addresses by the CMMU, which are then used to access the data cache, memory or other devices on the M-Bus. These bus interfaces, together with the data cache, have been designed to keep the user from having to develop a complex interface that is capable of feeding the processor efficiently.

The P-Bus interface supports 33 address lines, 32 bit data lines, and control lines such as Chip Select, 4 Data Byte Enables, Read/Write, Lock, and 2 REPLY lines. The bus protocol is synchronous and pipelined, optimized to supply the processor with either code or data at a peak rate of one word per cycle (80 M bytes/sec at 20MHz). For fault detection purposes, each P-Bus output (actually all outputs in the CMMU) has a comparator circuit that checks the signal on the pin against the one that is fed to its driver. If a discrepancy is detected, the ERR (error detect) signal is asserted one cycle later.

To support system fault tolerance, the P-Bus interface can be placed in the checker or shadow mode. In this mode, all output drivers are disabled, allowing a master CMMU to be coupled to one or more checker CMMUs. The checkers have access to the code and data streams and execute concurrently in lock step with the master, verifying every output signal but not driving any. If a checker detects a difference between the output of the master and its own internal signal, it asserts the ERR signal.

The M-Bus interface, described in detail in a later section, supports a synchronous 32 bit bus with multiplexed address and data and control signals to perform bus arbitration, specify Read/Write operations, establish exclusive use of global resources, inhibit external caching, signal bus errors, and provide initialization at Reset. The M-Bus interface works in conjunction with the cache control logic to automatically maintain cache coherency via an efficient bus

snooping protocol. This coherency logic insures that when a device attempts to access a memory location that the cache has a modified copy of, the cache will update memory before allowing the device to complete the access. Cache/memory coherency is a critical issue in multiprocessing systems, or in systems with Direct Memory Access devices.

The CMMU can be configured, controlled, and monitored by software through its register file, which consists of four categories of registers:

1. System interface registers: ID, Command, Status, Address, and Control registers.

The ID Register contains a 7 bit code that uniquely identifies the CMMU when its register file is accessed from the M-Bus. These ID bits are set when the CMMU comes out of reset via 7 external pins that are output-only during normal operation. The CMMU ID may be changed dynamically by writing a new code into this register.

The System Command Register allows software to initiate cache flushes and address probe operations. A probe of a logical address returns the corresponding physical address that is mapped to it, and all the protection/control bits pertaining to that address. Flushes and probes will begin after the appropriate command code is written into this register, either from the P-Bus or M-Bus.

The System Status Register reports M-Bus errors that occur during cache flushes, address probes, and memory updates caused by cache snoop hits (snoop copyback error). This register also reports the results of a probe, showing all the protection and control bits that pertain to the address being probed.

The System Address Register supplies addresses for the probe and cache flush commands, and to select set and word addresses for software accesses of the data cache via the cache diagnostic ports. The physical address of an address probe or of a faulted flush is also returned in this register.

The System Control Register allows software to enable parity checking on M-Bus READs, enable snooping of global data accesses on the M-Bus to maintain cache coherency, and select the priority or fairness protocol when the CMMU arbitrates for the M-Bus.

2. The Local Status and Address Registers are updated by the CMMU when a P-Bus transaction such as ld (load), st (store), or xmem (memory exchange) ends in a fault condition. The status register reports one of the fault codes: invalid segment fault, invalid page fault, supervisor violation, write-protect violation, or bus error. The address register contains the physical address of the location where the fault occurs, except in the case of a write-protect fault where no address will be reported. In this case the faulted address will be available in an exception register in the CPU.

3. Memory management registers: Supervisor and User Area Pointers, and BATC (Block Address Translation Cache) write ports. These registers form the basis of two alternate logical-to-physical address translation processes. Each of the two area pointers can be initialized by software to contain an area descriptor, part of which is the physical address of a segment descriptor table which is accessed during a hierarchical table walk (described in detail later) to perform an address translation. A Translation Enable bit in each of these area pointers, when cleared by software, causes the CMMU to bypass the translation process and treat logical addresses presented on the P-Bus as physical addresses.

Address mapping for blocks of 512K bytes of memory are stored in 8 BATC entries via the corresponding write ports. Each of these entries consists of a 14 bit CAM (Content Addressable Memory) containing a logical block address, a 13 bit RAM containing the corresponding physical block address, various control and protection bits pertaining to the block being mapped, and a Valid bit indicating whether the entry is valid. In addition, two hard-wired entries provide direct mapping for "control memory" which occupies the top 1M bytes of the total 4G bytes address range.

4. Cache Diagnostic Ports: 4 cache tag ports, 4 cache data ports, and one cache set status register. These ports allow software to access the various components of the cache as CMMU internal register accesses, mainly for diagnostic purposes.

Memory Management

The address space is divided logically into a supervisor and user space of 4G bytes each. The top 1M bytes of the supervisor space is control space reserved for mapping of control registers of M-Bus peripheral devices. The CMMU uses the S/U (supervisor/user) bit which is part of the logical address to select the appropriate space. These spaces can be mapped to a 4G bytes or smaller physical space, at the block (512K bytes), segment (4M bytes), or page (4K bytes) granularity.

The page mapping is implemented in a hierarchy of area, segment, and page descriptors. Each area and segment descriptor contains the physical address of a 4K byte table pointing to the next level descriptors. In the case of a page descriptor, the address points to a 4k physical memory space.

The descriptors also include protection/control bits for the portion of memory being mapped. Each of the pointers in the two area descriptors points to a segment table which contains up to 1024 valid segment descriptors. Each segment descriptor in turn points to a page table with up to 1024 valid page descriptors. Each page descriptor points to the physical page to which the logical address is mapped. Figure 3 shows the configuration of the three descriptors.

The memory management portion of the CMMU efficiently handles logical-to-physical address mapping described above via two ATCs (Address Translation Caches): a BATC (Block ATC) and a PATC (Page ATC) which functions in parallel. The BATC, which maps blocks of 512K bytes, is optimized for operating system uses, or can be used to map any user space memory that is mapped into contiguous 512 K byte blocks. The PATC consists of 56 entries, each consisting of the S/U bit and upper 20 bits of a logical address stored in CAM, the corresponding 20 bit physical address, and protection/control bits for a page of 4K bytes. These entries are formed and replaced by a hierarchical table walk automatically initiated when an ATC miss occurs. Figure 4 shows the format of the ATC entries.

When a memory access is to be performed (code fetch or data LOAD/

STORE; code fetches are the same as data LOADs from the CMMU perspective), the processor presents a logical address to the respective CMMU on the P-Bus address lines. The CMMU latches this address and if translation is enabled, performs simultaneous associative searches in both of the ATCs. In the BATC, the upper 14 bits (S/U and 13 MSB) of the incoming logical address is associatively compared to the logical address field of the 10 entries. If a match occurs in an entry, the corresponding physical block address is issued, concatenated with the remaining lower bits of the logical address to form the complete physical address of a word within that block, which is then used to access memory. In the PATC, a similar associative search is performed using the upper 20 bits of the logical address and the S/U bit, yielding the physical address of a 4K

byte page. The lower 12 bits of the logical address are concatenated with the page address to form the physical word address within that page. If there is an address match in both ATCs, the one in the BATC takes precedence.

If there is no match in either of the ATCs, the CMMU initiates a two-level table look up in memory to obtain the physical address as shown in Figure 5.

First, a segment descriptor is fetched from one of the segment tables. The physical address of this descriptor is formed by concatenating the segment table address given in the area pointer selected by the S/U bit and the 10 most significant bits of the logical address. The page descriptor is then fetched at the address given by the concatenation of the page table address in the segment descriptor and bits 12-23 of the logical address. Finally the physical address of the word to

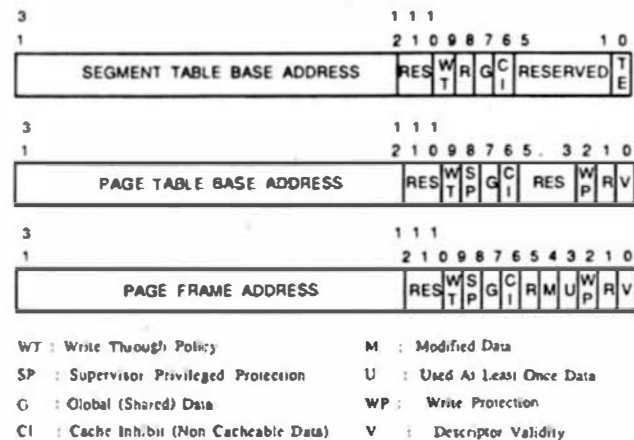


Figure 3 Area, Segment and Page Descriptors

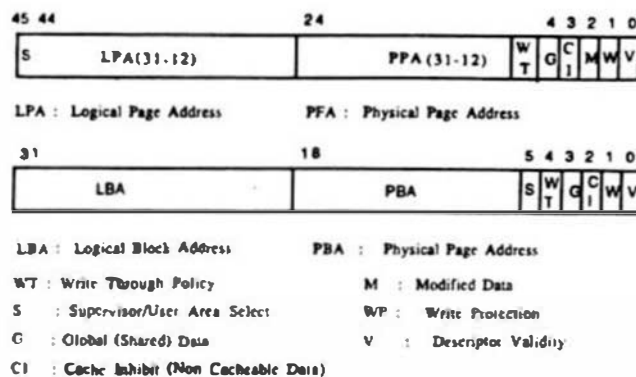


Figure 4 Address Translation Cache Entries

be accessed is formed by concatenating the physical page address in the page descriptor with the 12 LSB of the logical address. Basically, the two groups of 10 MSB of the logical addresses provide offsets into the segment and page tables, and the 12 LSB the offset into the physical page table. The top 20 bits of the physical address and logical address plus the S/U bit, together with all the protection/control bits accumulated from the descriptors are formed into a PATEntry and stored in the PATC, using a FIFO entry replacement policy. The physical address is concurrently presented to the cache or to memory to obtain the word requested in the P-Bus transaction.

DATA CACHE

The CMMU data cache has been designed to exploit temporal and spatial locality, providing 98% or better cache hit rate in most applications. The cache fast SRAM allows the CMMU to supply data to the processor at the sustained rate of one 32 bit word per cycle while there are cache hits.

The 16K bytes of high speed SRAM are organized as 256 sets of 4 lines each, with each line containing 4 32-bit words, with a 20 bit physical address tag, and valid and disable bits as shown in Figure 6. The two valid bits indicate the state of the cache line.

This design, taking advantage of the fact that the lower 12 bits of an address are the same for both the logical and physical address, achieves concurrency between address translations and data cache accesses: the cache is structured so that address bits 11-4 are decoded to select one of 256 sets, while bits 2 and 3 select one of 4 words in the set; this decoding can be initiated and even completed before the MMU has finished the address translation. Once the translation is done, the upper 20 bits of the physical address supplied by the MMU are associatively compared with the 4 physical address tags in the selected set, and a cache hit signaled if a match occurs. The data word is then

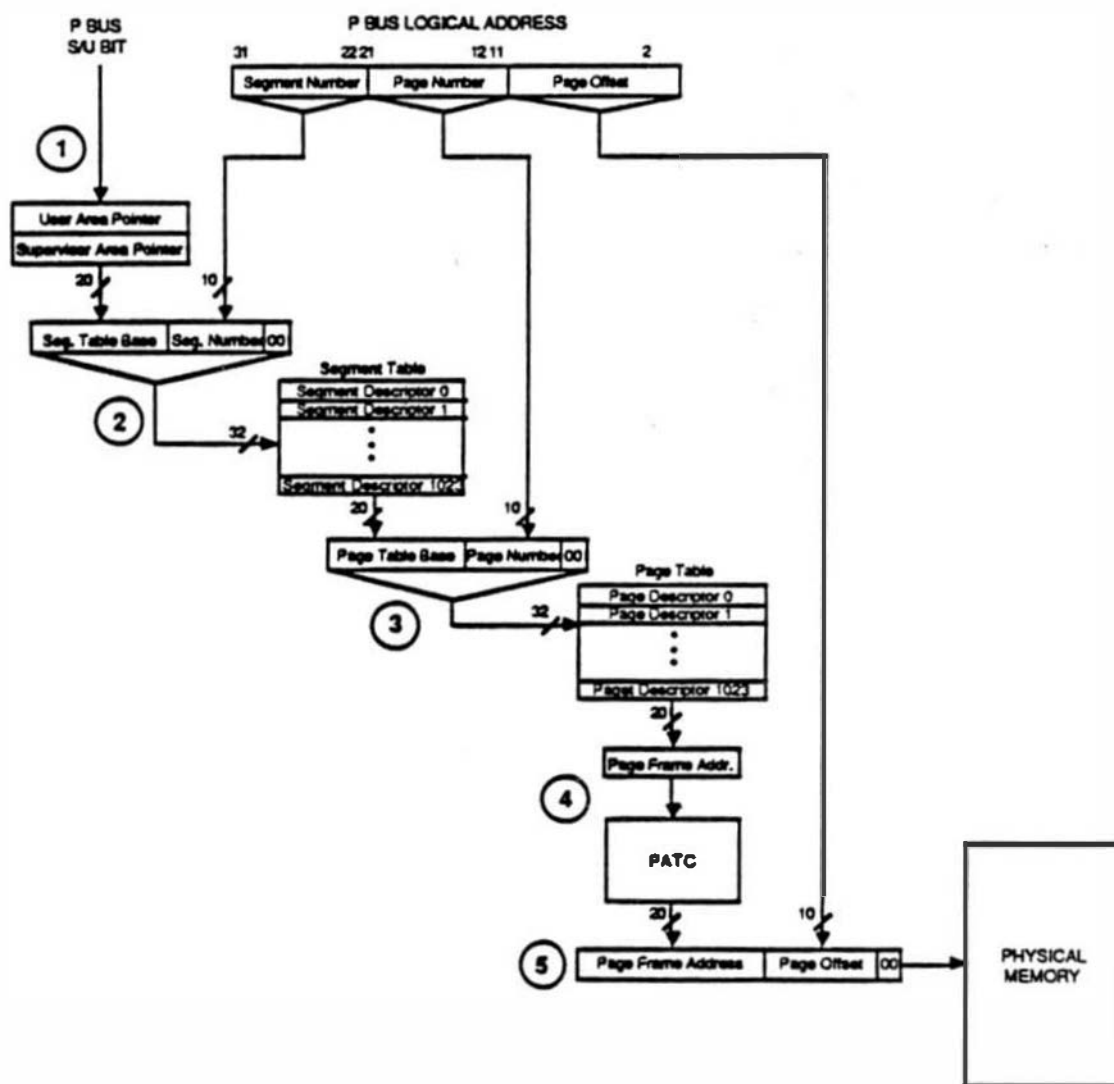


Figure 5 Two Level Table Walk

Telephone: (615) 842-4600

South East Media

OS-9, UniFLEX, FLEX, SK-DOS

Telex: 5106006630

ASSEMBLERS

- ASTRUK09** from S.E. Media -- A "Structured Assembler for the 6809" which requires the TSC Macro Assembler.
FLEX, SK-DOS, CCF - \$99.95
- Macro Assembler for TSC** - The FLEX, SK-DOS STANDARD Assembler.
Special -- CCF \$35.00; FLEX, SK-DOS \$50.00
- OSM Extended 6809 Macro Assembler** from Lloyd I/O. -- Provides local labels, Motorola S-records, and Intel Hex records; XREF. Generate OS-9 Memory modules under FLEX, SK-DOS.
FLEX, SK-DOS, CCF, OS-9 \$99.00
- Relocating Assembler/Linking Loader** from TSC. -- Use with many of the C and Pascal Compilers.
FLEX, SK-DOS, CCF \$150.00
- MACE**, by Graham Trotter from Windrush Micro Systems -- Co-Resident Editor and Assembler: fast interactive A.L. Programming for small to medium-sized Programs.
FLEX, SK-DOS, CCF - \$75.00
- XMACE** -- MACE w/Cross Assembler for 6800/1/2/3/8
FLEX, SK-DOS, CCF - \$98.00

DISASSEMBLERS

- SUPER SLEUTH** from Computer Systems Consultants Interactive Disassembler; extremely **POWERFUL!** Disk File Binary/ASCII Examine/Change, Absolute or FULL Disassembly. XREF Generator, Label "Name Changer", and Files of "Standard Label Names" for different Operating Systems.
Color Computer SS-50 Bus (all w/ A.L. Source)
CCD (32K Req'd) Object Only \$49.00
FLEX, SK-DOS \$99.00 - CCF Object Only \$50.00 UniFLEX \$100.00
CCF, with Source \$99.00 OS-9, \$101.00 - CCO, Object Only \$50.00
68010 SUPER SLEUTH - Similar to 8-Bit Version except written in "C".
68010 Disassembler \$100.00 FLEX, UniFLEX, UNIX, XENIX, MS-DOS, SK-DOS, OS-9
OS-9/68K Object Only \$100.00 or with Source \$200.00
- DYNAMITE+** -- Excellent standard "Batch Mode" Disassembler. Includes XREF Generator and "Standard Label" Files. Special OS-9 options with OS-9 Version.
CCF, Object Only \$100.00 - CCO, Object Only \$59.95
FLEX, SK-DOS, Object Only \$100.00 - OS-9, Object Only \$150.00
UniFLEX Object Only \$300.00

CROSS ASSEMBLERS

- CROSS ASSEMBLERS** from Computer System Consultants -- Supports 1802/5, Z-80, 6800/1/2/3/8/11/HC11, 6804, 6805/HC05/ 146805, 6809/0Q01, 6502 family, 8080/5, 8020/1/2/3/5/39/ 40/48/C48/49/C49/50/ 8748/49, 8031/51/8751, 32000 and 68000/68010 Systems. Assembler and Listing formats same as target CPU's format. Produces machine independent Motorola S-Text. Includes Macro Pre-Processor. Written in "C". 68000 or 6809 *Macintosh, *Atari, FLEX, CCF, UniFLEX, OS-9, XENIX, UNIX, MS-DOS, SK-DOS
any object or source each - \$50.00
any 3 object or source - \$100.00
Set of ALL object \$200.00 - with source \$500.00
- XASM Cross Assemblers** for FLEX, SK-DOS from S.E. MEDIA -- This set of 6800/1/2/3/5/8, 6301, 6502, 8080/5, and Z80 Cross Assemblers uses the familiar TSC Macro Assembler Command Line and Source Code format. Assembler options, etc., in providing code for target CPU's.
Complete set, FLEX, SK-DOS only - \$150.00

CRASMB from LLOYD I/O -- Supports Motorola's, Intel's, Zilog's, and other's CPU syntax for these 8-Bit microprocessors: 6800, 6801, 6303, 6804, 6805, 6809, 6811 (all varieties); 6502, 1802/5, 8048 family, 8051 family, 8080/85, Z8, Z80, and TMS-7000 family. Has MACROS, Local Labels, Label X-REF, Label Length to 30 Chars. Object code formats: Motorola S-Records (text), Intel HEX-Records (text), OS-9 (binary), and FLEX, SK-DOS (binary). Written in Assembler ... e.g. **Very Fast.**

CPU TYPE - Price each:

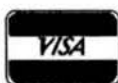
For:	MOTOROLA	INTEL	OTHER COMPLETE SET
FLEX9	\$150	\$150	\$399
SK-DOS	\$150	\$150	\$399
OS-9/6809	\$150	\$150	\$399
OS-9/68K	-----	-----	\$432

CRASMB 16.32 from LLOYD I/O -- Supports Motorola's 68000, and has same features as the 8 bit version. OS9/68K Object code Format allows this cross assembler to be used in developing your programs for OS-9/68K on your OS-9/6809 computer.
FLEX, SK-DOS, CCF, OS-9/6809 \$249.00

COMMUNICATIONS

- C-MODEM** Telecommunications Program from Computer Systems Consultants, Inc. -- Menu-Driven; supports Dumb-Terminal Mode, Upload and Download in non-protocol mode, and the CP/M "Modem7" Christensen protocol mode to enable communication capabilities for almost any requirement. Written in "C".
FLEX, SK-DOS, CCF, OS-9, UniFLEX, UNIX, XENIX, MS-DOS, with Source \$100.00 - without Source \$50.00
- X-TALK** from S.E. Media - X-TALK consists of two disks and a special cable, the hookup enables a 6809 SWTPC computer to dump UniFLEX files directly to the UniFLEX MUSTANG-020. This is the ONLY currently available method to transfer SWTPC 6809 UniFLEX files to a 68000 UniFLEX system. Gimix 6809 users may dump a 6809 UniFLEX file to a 6809 UniFLEX five inch disk and it is readable by the MUSTANG-020. The cable is specially prepared with internal connections to match the non-standard SWTPC SO/9 I/O Db25 connectors. A special SWTPC S+ cable set is also available. Users should specify which SWTPC system he/she wishes to communicate with the MUSTANG-020. The X-TALK software is furnished on two disks. One eight inch disk contains S.E. Media modem program C-MODEM (6809) and the other disk is a MUSTANG-020 five inch disk with C-MODEM (68020). Text and binary files may be directly transferred between the two systems. The C-MODEM programs are unaltered and perform as excellent modem programs also. X-TALK can be purchased with or without the special cables, but this special price is available to registered MUSTANG-020 users only.
X-TALK Complete (cable, 2 disks) \$99.95
X-TALK Software (2 disks only) \$69.95
X-TALK with C-MODEM Source \$149.95
- XDATA** from S.E. Media - A COMMUNICATION Package for the UniFLEX Operating System. Use with CP/M, Main Frames, other UniFLEX Systems, etc. Verifies Transmission using checksum or CRC; Re-Transmits bad blocks, etc.
UniFLEX - \$299.99

Availability Legend:
 O = OS-9, S = SK-DOS
 F = FLEX, U = UniFLEX
 CC = Color Computer OS-9
 CCP = Color Computer FLEX



South East Media
 5900 Cassandra Smith Rd. - Hixson, TN. 37343



**** Shipping ****
 Add 2% U.S.A. (min. \$2.00)
 Foreign Surface Add 5%
 Foreign Airmail Add 10%
 Or C.O.D. Shipping Only

*OS-9 is a Trademark of Microware and Motorola. *FLEX and UniFLEX are Trademarks of Technical Systems Consultants. *SK-DOS is a Trademark of Star-K Software Systems Corp.

Telephone: (615) 842-4600

South East Media

OS-9, UniFLEX, FLEX, SK-DOS

Telex: 5106006630

PROGRAMMING LANGUAGES

PL/9 from Windrush Micro Systems -- By Graham Todd. A combination Editor Compiler Debugger. Direct source-to-object compilation delivering fast, compact, re-entrant, ROM-able, PIC, 8 & 16-bit Integers & 6-digit Real numbers for all real-world problems. Direct control over ALL System resources, including interrupts. Comprehensive library support; simple Machine Code interface; step-by-step tracer for instant debugging. 500+ page Manual with tutorial guide.

FLEX, SK-DOS, CCF - \$198.00

PASC from S.E. Media - A FLEX9, SK-DOS Compiler with a definite Pascal "flavor". Anyone with a bit of Pascal experience should be able to begin using PASC to good effect in short order. The PASC package comes complete with three sample programs: ED (a syntax or structure editor), EDITOR (a simple, public domain, screen editor) and CHESS (a simple chess program). The PASC package comes complete with source (written in PASC) and documentation.

FLEX, SK-DOS \$95.00

WHIMSICAL from S.E. MEDIA Now supports Real Numbers. "Structured Programming" WITHOUT losing the Speed and Control of Assembly Language! Single-pass Compiler features unified, user-defined I/O; produces ROMable Code; Procedures and Modules (including pre-compiled Modules); many "Types" up to 32 bit Integers, 6-digit Real Numbers, unlimited sized Arrays (vectors only); Interrupt handling; long Variable Names; Variable Initialization; Include directive; Conditional compiling; direct Code insertion; control of the Stack Pointer, etc. Run-Time subroutines inserted as called during compilation. Normally produces 10% less code than PL/9.

FLEX, SK-DOS and CCF - \$195.00

KANSAS CITY BASIC from S.E. Media - Basic for Color Computer OS-9 with many new commands and sub-functions added. A full implementation of the IF-THEN-ELSE logic is included, allowing nesting to 255 levels. Strings are supported and a subset of the usual string functions such as LEFTS, RIGHTS, MIDS, STRINGS, etc. are included. Variables are dynamically allocated. Also included are additional features such as Peek and Poke. A must for any Color Computer user running OS-9.

CoCo OS-9 \$39.95

C Compiler from Windrush Micro Systems by James McCosh. Full C for FLEX, SK-DOS except bit-fields, including an Assembler. Requires the TSC Relocating Assembler if user desires to implement his own Libraries.

FLEX, SK-DOS, CCF - \$295.00

C Compiler from Introl -- Full C except Doubles and Bit Fields, streamlined for the 6809. Reliable Compiler; FAST, efficient Code. More UNIX Compatible than most.

FLEX, SK-DOS, CCF, OS-9 (Level II ONLY), UniFLEX - \$575.00

PASCAL Compiler from Lucidata -- ISO Based P-Code Compiler.

Designed especially for Microcomputer Systems. Allows linkage to Assembler Code for maximum flexibility.

FLEX, SK-DOS and CCF - \$190.00

OmegaSoft PASCAL from Certified Software -- Extended Pascal for systems and real-time programming.

Native 68000/68020 Compiler, \$575 for base package, options available. For OS-9/68000 and PDOS host system.

6809 Cross Compiler (OS-9/68000 host) \$700 for complete package.

KBASIC - from S.E. MEDIA - A "Native Code" BASIC Compiler which is now Fully TSC XBASIC compatible. The compiler compiles to Assembly Language Source Code. A NEW, streamlined, Assembler is now included allowing the assembly of LARGE Compiled K-BASIC Programs. Conditional assembly reduces Run-time package.

FLEX, SK-DOS, CCF, OS-9 Compiler/Assembler \$99.00

CRUNCH COBOL from S.E. MEDIA -- Supports large subset of ANSI Level 1 COBOL with many of the useful Level 2 features. Full FLEX, SK-DOS File Structures, including Random Files and the ability to process Keyed Files. Segment and link large programs at runtime, or implemented as a set of overlays. The System requires 56K and CAN be run with a single Disk System. A very popular product.

FLEX, SK-DOS, CCF - \$99.95

FORTH from Stearns Electronics -- A CoCo FORTH Programming Language. Tailored to the CoCo! Supplied on Tape, transferable to disk. Written in FAST ML. Many CoCo functions (Graphics, Sound, etc.). Includes an Editor, Trace, etc. Provides CPU Carry Flag accessibility, Fast Task Multiplexing, Clean Interrupt Handling, etc. for the "Pro". Excellent "Learning" tool!

Color Computer ONLY - \$58.95

FORTHBUILDER is a stand-alone target compiler (crosscompiler) for producing custom Forth systems and application programs. All of the 83-standard defining words and control structures are recognized by FORTHBUILDER.

FORTHBUILDER is designed to behave as much as possible like a resident Forth interpreter/compiler, so that most of the established techniques for writing Forth code can be used without change. Like compilers for other languages, FORTHBUILDER can operate in "batch mode".

The compiler recognizes and emulates target names defined by CONSTANT or VARIABLE and is readily extended with "compile-time" definitions to emulate specific target words.

FORTHBUILDER is supplied as an executable command file configured for a specific host system and target processor. Object code produced from the accompanying model source code is royalty-free to licensed users.

FLEX, CCF, SK-DOS - \$99.95

EDITORS & WORD PROCESSING

JUST from S.E. Media -- Text Formatter developed by Ron Anderson; for Dot Matrix Printers, provides many unique features. Output "Formatted" Text to the Display. Use the FPRINT.COM supplied for producing multiple copies of the "Formatted" Text on the Printer INCLUDING IMBEDDED PRINTER COMMANDS (very useful at other times also, and worth the price of the program by itself). "User Configurable" for adapting to other Printers (comes set up for Epson MX-80 with Graftrax); up to ten (10) imbedded "Printer Control Commands". Compensates for a "Double Width" printed line. Includes the normal line width, margin, indent, paragraph, space, vertical skip lines, page length, page numbering, centering, fill, justification, etc.

Use with PAT or any other editor.

* Now supplied as a two disk set:
Disk #1: JUST2.COM object file,
JUST2.TXT PL9 source: FLEX, SK-DOS - CCF
Disk #2: JUSTSC object and source in C:
FLEX, SK-DOS, OS-9, CCF

* Now supplied as a two disk set:

Disk #1: JUST2.COM object file,

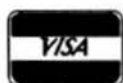
JUST2.TXT PL9 source: FLEX, SK-DOS - CCF

Disk #2: JUSTSC object and source in C:

FLEX, SK-DOS, OS-9, CCF

The JTSC and regular JUST C source are two separate programs. JTSC compiles to a version that expects TSC Word Processor type commands, (.pp .sp .oe etc.) Great for your older text files. The C

Availability Legend
O = OS-9, S = SK-DOS
F = FLEX, U = UniFLEX
CO9 = Color Computer OS-9
CCF = Color Computer FLEX



South East Media
5900 Cassandra Smith Rd. - Hixson, TN 37343



** Shipping **
Add 2% U.S.A. (min. \$2.50)
Foreign Surface Add 5%
Foreign Airmail Add 10%
Or C.O.D. Shipping Only

*OS-9 is a Trademark of Microware and Motorola. *FLEX and UniFLEX are Trademarks of Technical Systems Consultants. *SK-DOS is a Trademark of Star-K Software Systems Corp.

Telephone: (615) 842-4600

South East Media

OS-9, UniFLEX, FLEX, SK-DOS

Telex: 5106006630

source compiles to a standard syntax JUST.CMD object file. Using JUST syntax (.p, .u, .y, etc.) With all JUST functions plus several additional printer formatting functions. Reference the JUSTSC C source. For those wanting an excellent BUDGET PRICED word processor, with features none of the others have. This is it!

Disk (1) - PL9 FLEX only. FLEX, SK-DOS & CCF - \$49.95
Disk Set (2) - FLEX, SK-DOS & CCF & OS-9 (C version) - \$69.95
OS-9 68K000 complete with Source - \$79.95

PAT from S.E. Media - A full feature screen oriented TEXT EDITOR with all the best of "PIE™". For those who swore by and loved only PIE, this is for you! All PIE features & much more! Too many features to list. And if you don't like these, change or add your own. PL-9 source furnished. "C" source available soon. Easily configured to your CRT, with special config section.

Regular FLEX, SK-DOS \$129.50

* SPECIAL INTRODUCTION OFFER * \$79.95

SPECIAL PAT/JUST COMBO (with source)

FLEX, SK-DOS \$99.95

OS-9 68K Version \$229.00

SPECIAL PAT/JUST COMBO 68K \$249.00

Note: JUST in "C" source available for OS-9

CEDRIC from S.E. Media - A screen oriented TEXT EDITOR with availability of 'MENU' aid. Macro definitions, configurable 'permanent definable MACROS' - all standard features and the fastest 'global' functions in the west. A simple, automatic terminal config program makes this a real 'no hassle' product. Only 6K in size, leaving the average system over 165 sectors for text buffer - approx. 14,000 plus of free memory! Extra fine for programming as well as text.

FLEX, SK-DOS \$69.95

BAS-EDIT from S.E. Media - A TSC BASIC or XBASIC screen editor. Appended to BASIC or XBASIC, BAS-EDIT is transparent to normal BASIC/XBASIC operation. Allows editing while in BASIC/XBASIC. Supports the following functions: OVERLAY, INSERT and DUP LINE. Make editing BASIC/XBASIC programs SIMPLE! A GREAT time and effort saver. Programmers love it! NO more retyping entire lines, etc. Complete with over 25 different CRT terminal configuration overlays.

FLEX, CCF, SK-DOS \$39.95

SCREEDITOR III from Windrush Micro Systems -- Powerful Screen-Oriented Editor/Word Processor. Almost 50 different commands; over 300 pages of Documentation with Tutorial. Features Multi-Column display and editing, "decimal align" columns (AND add them up automatically), multiple keystroke macros, even/odd page headers and footers, imbedded printer control codes, all justifications, "help" support, store common command series on disk, etc. Use supplied "set-up", or remap the keyboard to your needs. Except for proportional printing, this package will DO IT ALL!

6800 or 6809 FLEX, SK-DOS or SSB-DOS, OS-9 - \$175.00

SPELLB "Computer Dictionary" from S.E. Media -- OVER 150,000 words! Look up a word from within your Editor or Word Processor (with the SPH.CMD Utility which operates in the FLEX, SK-DOS UCS). Or check and update the Text after entry; ADD WORDS to the Dictionary. "Flag" questionable words in the Text, "View a word in context" before changing or ignoring, etc. SPELLB first checks a "Common Word Dictionary", then the normal Dictionary, then a "Personal Word List", and finally, any "Special Word List" you may have specified. SPELLB also allows the use of Small Disk Storage systems.

FLEX, SK-DOS and CCF - \$129.95

STYLO-GRAPH from Great Plains Computer Co. -- A full-screen oriented WORD PROCESSOR -- (uses the 51 x 24 Display Screens on CoCo FLEX/SK-DOS, or PBJ Wordpak). Full screen display and editing; supports the Daisy Wheel proportional printers.

NEW PRICES 6809 CCF and CCO - \$99.95,

FLEX, SK-DOS or OS-9 - \$179.95, UniFLEX - \$299.95

STYLO-SPELL from Great Plains Computer Co. -- Fast Computer Dictionary. Complements Stylograph.

NEW PRICES 6809 CCF and CCO - \$69.95,

FLEX, SK-DOS or OS-9 - \$99.95, UniFLEX - \$149.95

STYLO-MERGE from Great Plains Computer Co. -- Merge Mailing List to "Form" Letters, Print multiple Files, etc., through Stylo.

NEW PRICES 6809 CCF and CCO - \$59.95,

FLEX, SK-DOS or OS-9 - \$79.95, UniFLEX - \$129.95

STYLO-PAK -- Graph + Spell + Merge Package Deal!!!

FLEX, SK-DOS or OS-9 - \$329.95, UniFLEX - \$549.95

OS-9 68000 \$695.00

DATABASE ACCOUNTING

XDMS from Westchester Applied Business Systems

FOR 6809 FLEX or SK-DOS (5/8")

Up to 32 groups/fields per record! Up to 12 character file names! Up to 1024 byte records! User defined screen and print control! Process files! Form files! Conditional execution! Process chaining! Upward/Downward file linking! File joining! Random file virtual paging! Built in utilities! Built in text line editor! Fully session oriented! Enhanced forms! Boldface, Double width, italics and Underline supported! Written in compact structured assembler! Integrated for FAST execution!

XDMS-IV Data Management System

XDMS-IV is a brand new approach to data management. It not only permits users to describe, enter and retrieve data, but also to process entire files producing customized reports, screen displays and file output. Processing can consist of any of a set of standard high level functions including record and field selection, sorting and aggregation, lookups in other files, special processing of record subsets, custom report formatting, totaling and subtotaling, and presentation of up to three related files as a "database" on user defined output reports.

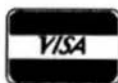
POWERFUL COMMANDS!

XDMS-IV combines the functionality of many popular DBMS software systems with a new easy to use command set into a single integrated package. We've included many new features and commands including a set of general file utilities. The processing commands are Input-Process-Output (IPO) which allows almost instant implementation of a process design.

SESSION ORIENTED!

XDMS-IV is session oriented. Enter "XDMS" and you are in instant command of all the features. No more waiting for a command to load in from disk! Many commands are immediate, such as CREATE (file definition), UPDATE (file editor), PURGE and DELETE (utilities). Others are process commands which are used to create a user process which is executed with a RUN command. Either may be entered into a "process" file which is executed by an EXECUTE statement. Processes may execute other processes, or themselves, either conditionally or unconditionally. Menus and screen prompts are easily coded, and entire user applications can be run without ever leaving XDMS-IV.

Availability Legends
O = OS-9, S = SK-DOS
F = FLEX, U = UniFLEX
CCF = Color Computer OS-9
CCF = Color Computer FLEX



South East Media
5900 Cassandra Smith Rd. - Hixson, Tn. 37343



** Shipping **
Add 2% U.S.A. (min. \$2.50)
Foreign Surcharge Add 5%
Foreign Airmail Add 10%
Or C.O.D. \$ Shipping Only

*OS-9 is a Trademark of Microware and Motorola. *FLEX and UniFLEX are Trademarks of Technical Systems Consultants. *SK-DOS is a Trademark of Star-K Software Systems Corp.

Telephone: (615) 842-4600

South East Media

OS-9, UniFLEX, FLEX, SK-DOS

Telex: 5106006630

IT'S EASY TO USE!

XDMS-IV keeps data management simple! Rather than design a complex DBMS which hides the true nature of the data, we kept XDMS-IV file oriented. The user view of data relationships is presented in reports and screen output, while the actual data resides in easy to maintain files. This aspect permits customized presentation and reports without complex redefinition of the database files and structure. XDMS-IV may be used for a wide range of applications from simple record management systems (addresses, inventory ...) to integrated database systems (order entry, accounting...)

The possibilities are unlimited...

FOR 6809 FLEX or SK-DOS(5 1/8" Disk) **\$249.95**

UTILITIES

Basic09 XRef from S.E. Media -- This Basic09 Cross Reference Utility is a Basic09 Program which will produce a "pretty printed" listing with each line numbered, followed by a complete cross referenced listing of all variables, external procedures, and line numbers called. Also includes a Program List Utility which outputs a fast "pretty printed" listing with line numbers. Requires Basic09 or RunB.

OS-9 & CCO object only -- \$39.95; with Source -- \$79.95

BTree Routines - Complete set of routines to allow simple implementation of keyed files - for your programs - running under Basic09. A real time saver and should be a part of every serious programmers tool-box.

OS-9 & CCO object only - \$89.95

Lucidata PASCAL UTILITIES (Requires Pascal ver 3)

XREF -- produce a Cross Reference Listing of any text; oriented to Pascal Source.

INCLUDE -- Include other Files in a Source Text, including Binary - unlimited nesting.

PROFILER -- provides an Indented, Numbered, "Structogram" of a Pascal Source Text File; view the overall structure of large programs, program integrity, etc. Supplied in Pascal Source Code; requires compilation.

FLEX, SK-DOS, CCF ... EACH 5" - \$40.00, 8" - \$50.00

DUB from S.E. Media -- A UniFLEX BASIC decompiler Re-Create a Source Listing from UniFLEX Compiled basic Programs. Works with ALL Versions of 6809 UniFLEX basic.

UniFLEX - \$219.95

LOW COST PROGRAM KITS from Southeast Media The following kits are available for FLEX, SK-DOS on either 5" or 8" Disk.

1. BASIC TOOL-CHEST \$29.95

BLISTER.CMD: pretty printer

LINEXREF.BAS: line cross-referencer

REMPAC.BAS, SPCPAC.BAS, COMPAC.BAS:

remove superfluous code

STRIP.BAS: superfluous line-numbers stripper

2. FLEX, SK-DOS UTILITIES KIT \$39.99

CATS. CMD: alphabetically-sorted directory listing

CATD.CMD: date-sorted directory listing

COPYSORT.CMD: file copy, alphabetically

COPYDATE.CMD: file copy, by date-order

FILEDATE.CMD: change file creation date

INFO.CMD (& **INFOGMX.CMD**): tells disk attributes & contents

RELINK.CMD (& **RELINK82**): re-orders fragmented free chain

RESQ.CMD: undeletes (recovers) a deleted file

SECTORS.CMD: show sector order in free chain

XL.CMD: super text lister

3. ASSEMBLERS/DISASSEMBLERS UTILITIES \$39.95

LINEFEED.CMD: 'modularise' disassembler output

MATH.CMD: decimal, hex, binary, octal conversions & tables

SKIP.CMD: column stripper

4. WORD - PROCESSOR SUPPORT UTILITIES \$49.95

FULLSTOP.CMD: checks for capitalization

BSTYCT.BAS (.BAC): Stylo to dot-matrix printer

NECPRI.CMD: Stylo to dot-matrix printer filter code

5. UTILITIES FOR INDEXING \$49.95

MENU.BAS: selects required program from list below

INDEX.BAC: word index

PHRASES.BAC: phrase index

CONTENT.BAC: table of contents

INDXSORT.BAC: fast alphabetic sort routine

FORMATER.BAC: produces a 2-column formatted index

APPEND.BAC: append any number of files

CHAR.BIN: line reader

BASIC09 TOOLS consist of 21 subroutines for Basic09.

6 were written in C Language and the remainder in assembly.

All the routines are compiled down to native machine code which makes them fast and compact.

1. **CFILL** -- fills a string with characters
2. **DPEEK** -- Double peek
3. **DPOKE** -- Double poke
4. **FPOS** -- Current file position
5. **FSIZE** -- File size
6. **FTRIM** -- removes leading spaces from a string
7. **GETPR** -- returns the current process ID
8. **GETOPT** -- gets 32 byte option section
9. **GETUSR** -- gets the user ID
10. **GTIME** -- gets the time
11. **INSERT** -- insert a string into another
12. **LOWER** -- converts a string into lowercase
13. **READY** -- Checks for available input
14. **SETPRIOR** -- changes a process priority
15. **SETUSR** -- changes the user ID
16. **SETOPT** -- set 32 byte option packet
17. **STIME** -- sets the time
18. **SPACE** -- adds spaces to a string
19. **SWAP** -- swaps any two variables
20. **SYSCALL** -- system call
21. **UPPER** -- converts a string to uppercase

For OS-9 - \$44.95 - Includes Source Code

SOFTTOOLS

The following programs are included in object form for immediate application. PL/9 source code available for customization.

READ-ME Complete instructions for initial set-up and operation. Can even

be printed out with the included text processor.

CONFIG one time system configuration.

CHANGE changes words, characters, etc. globally to any text type file.

CLEANTXT converts text files to standard FLEX, SK-DOS files.

COMMON compare two text files and reports differences.

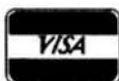
COMPARE another check file that reports mis-matched lines.

CONCAT similar to FLEX, SK-DOS append but can also list files to screen.

DOCUMENT for PL/9 source files. Very useful in examining parameter

passing aspects of procedures.

Availability Legend:
O = OS-9, S = SK-DOS
F = FLEX, U = UniFLEX
CO = Color Computer OS-9
CCF = Color Computer FLEX



South East Media

5900 Cassandra Smith Rd. - Hixson, TN. 37343



**** Shipping ****
Add 2% U.S.A. (min. \$2.50)
Foreign Surface Add 5%
Foreign Airmail Add 10%
Or C.O.D. Shipping Only

*OS-9 is a Trademark of Microware and Motorola. *FLEX and UniFLEX are Trademarks of Technical Systems Consultants. *SK-DOS is a Trademark of Star-K Software Systems Corp.

Telephone: (615) 842-4600

South East Media

Telex: 5106006630

OS-9, UniFLEX, FLEX, SK-DOS

ECHO echos to either screen or file.

FIND an improved find command with "pattern" matching and wildcards. Very useful.

HEX dumps files in both hex and ASCII.

INCLUDE a file copy program that will accept "includes" of other disk files.

KWIC allows rotating each word, on each line to the beginning. Very useful in a sort program, etc.

LISTDIR a directory listing program. Not super, but better than CAT.

MEMSORT a high-speed text file sorter. Up to 10 fields may be sorted. Very fast. Very useful.

MULTICOL width of page, number of columns may be specified. A MUST!

PAGE similar to LIST but allows for a page header, page width and depth. Adjust for CRT screen or printer as set up by CONFIG. A very smart print driver. Allows printer control commands.

REMOVE a fast file deleter. Careful, no prompts issued. Zap, and its gone!

SCREEN a screen listing utility. Word wraps text to fit screen. Screen depth may be altered at run time.

SORT a super version of MEMSORT. Ascending/descending order, up to 10 keys, case over-ride, sort on nth word and sort on characters if file is small enough, sorts in RAM. If large file, sort is constrained to size of your largest disk capacity.

TPROC a small but nice text formatter. This is a complete formatter and has functions not found in other formatters.

TRANSLIT sorts a file by x keyfields. Checks for duplications. Up to 10 key files may be used.

UNROTATE used with KWIC this program reads an input file and unfolds it a line at a time. If the file has been sorted each word will be presented in sequence.

WC a word count utility. Can count words, characters or lines.

NOTE: this set of utilities consists of 6 5-1/4" disks or 2 8" disks, with source (PL9). 3 5-1/4" disks or 1 8" disk without source.

Complete set SPECIAL INTRO PRICE:

5-1/4" with source FLEX or SK-DOS - \$129.95

without source - \$79.95

8" with source - \$79.95 - without source \$49.95

FULL SCREEN FORMS DISPLAY from Computer Systems Consultants -

- TSC Extended BASIC program supports any Serial Terminal with Cursor Control or Memory-Mapped Video Displays; substantially extends the capabilities of the Program Designer by providing a table-driven method of describing and using Full Screen Displays.

FLEX, SK-DOS and CCF, UniFLEX - \$25.00, with Source - \$50.00

SOLVE from S.E. Media - OS-9 Levels I and II only. A Symbolic Object/Logic Verification & Examine debugger. Including inline debugging, disassemble and assemble. SOLVE IS THE MOST COMPLETE DEBUGGER we have seen for the 6809 OS-9 series! SOLVE does it all! With a rich selection of monitor, assembler, disassembler, environmental, execution and other miscellaneous commands, SOLVE is the MOST POWERFUL tool-kit item you can own! Yet, SOLVE is simple to use! With complete documentation, a map! Everyone who has ordered this package has raved! See review - 68 Micro Journal - December 1985. No "blind" debugging here, full screen displays, rich and complete in information presented. Since review in 68 Micro Journal, this is our fastest mover!

Levels I & II only - OS-9 \$69.95

DISK UTILITIES

OS-9 VDisk from S.E. Media -- For Level I only. Use the Extended Memory capability of your SWTPC or Gimix CPU card (or similar format DAT) for FAST Program Compiles, CMD execution, high speed inter-process communications (without pipe buffers), etc. - SAVE that System Memory. Virtual Disk size is variable in 4K increments up to 960K. Some Assembly Required.

Level I OS-9 object \$79.95; with Source \$149.95

O-F from S.E. Media -- Written in BASIC09 (with Source), includes:

REFORMAT, a BASIC09 Program that reformats a chosen amount of an OS-9 disk to FLEX, SK-DOS Format so it can be used normally by FLEX, SK-DOS; and FLEX, a BASIC09 Program that does the actual read or write function to the special O-F Transfer Disk; user-friendly menu driven. Read the FLEX, SK-DOS Directory, Delete FLEX, SK-DOS Files, Copy both directions, etc. FLEX, SK-DOS use the special disk just like any other FLEX, SK-DOS disk

OS-9 - 6809/68000 \$79.95

LSORT from S.E. Media - A SORT/MERGE package for OS-9 (Level I & II only). Sorts records with fixed lengths or variable lengths. Allows for either ascending or descending sort. Sorting can be done in either ASCII sequence or alternate collating sequence. Right, left or no justification of data fields available. LSORT includes a full set of comments and errors messages.

OS-9 \$85.00

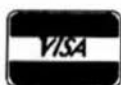
HIER from S.E. Media - HIER is a modern hierarchal storage system for users under FLEX, SK-DOS. It answers the needs of those who have hard disk capabilities on their systems, or many files on one disk - any size. Using HIER a regular (any) FLEX, SK-DOS disk (8" - 5" hard disk) can have sub directories. By this method the problems of assigning unique names to files is less burdensome. Different files with the exact same name may be on the same disk, as long as they are in different directories. For the winchester user this becomes a must. Sub-directories are the modern day solution that all current large systems use. Each directory looks to FLEX, SK-DOS like a regular file, except they have the extension '.DIR'. A full set of directory handling programs are included, making the operation of HIER simple and straightforward. A special install package is included to install HIER to your particular version of FLEX, SK-DOS. Some assembly required. Install indicates each byte or reference change needed. Typically - 6 byte changes in source (furnished) and one assembly of HIER is all that is required. No programming required!

FLEX - SK-DOS \$79.95

COPYMULT from S.E. Media -- Copy LARGE Disks to several smaller disks. FLEX, SK-DOS utilities allow the backup of ANY size disk to any SMALLER size diskettes (Hard Disk to Floppies, 8" to 5", etc.) by simply inserting diskettes as requested by COPYMULT. No fooling with directory deletions, etc. COPYMULT.CMD understands normal "copy" syntax and keeps up with files copied by maintaining directories for both host and receiving disk system. Also includes BACKUP.CMD to download any size "random" type file; RESTORE.CMD to restructure copied "random" files for copying, or recopying back to the host system; and FREELINK.CMD as a "bonus" utility that "relinks" the free chain of floppy or hard disk, eliminating fragmentation.

Completely documented Assembly Language Source files included. ALL 4 Programs (FLEX, SK-DOS, 8" or 5") \$99.50

Availability Legend
O = OS-9, S = SK-DOS
F = FLEX, U = UniFLEX
CC8 = Color Computer OS-9
CCF = Color Computer FLEX



South East Media

5900 Cassandra Smith Rd. - Hixson, TN. 37343



** Shipping **
Add 2% U.S.A. (incl. \$2.50)
Foreign Surface Add 5%
Foreign Airmail Add 10%
Or C.O.D. Shipping Only

*OS-9 is a Trademark of Microware and Motorola. *FLEX and UniFLEX are Trademarks of Technical Systems Consultants. *SK-DOS is a Trademark of Star-K Software Systems Corp.

Telephone: (615) 842-4600

South East Media

OS-9, UniFLEX, FLEX, SK-DOS

Telex: 5106006630

COPYCAT from Lucidata -- *Pascal NOT required.* Allows reading TSC Mini-FLEX, SK-DOS, SSB-DOS68, and Digital Research CP/M Disks while operating under SK-DOS, FLEX1.0, FLEX 2.0, or FLEX 9.0 with 6800 or 6809 Systems. COPYCAT will not perform miracles, but, between the program and the manual, you stand a good chance of accomplishing a transfer. Also includes some Utilities to help out. Programs supplied in Modular Source Code (Assembly Language) to help solve unusual problems.

FLEX, SK-DOS and CCF 5" - \$50.00 FLEX, SK-DOS 8" - \$65.00

VIRTUAL TERMINAL from S.E. Media - Allows one terminal to do the work of several. The user may start as many as eight tasks on one terminal, under *VIRTUAL TERMINAL* and switch back and forth between tasks at will. No need to exit each one; just jump back and forth. Complete with configuration program. The best way to keep up with those background programs.

6809 OS-9 & CCO - object only - \$49.95

FLEX, SK-DOS DISK UTILITIES from Computer Systems Consultants -- Eight (8) different Assembly Language (with Source Code) FLEX, SK-DOS Utilities for every FLEX, SK-DOS Users Toolbox: Copy a File with CRC Errors; Test Disk for errors; Compare two Disks; a fast Disk Backup Program; Edit Disk Sector; Linearize Free Chain on the Disk; print Disk Identification; and Sort and Replace the Disk Directory (in sorted order). -- PLUS -- Ten X BASIC Programs including: A BASIC Resequencer with EXTRAS over "RENUM" like check for missing label definitions, processes Disk to Disk instead of in Memory, etc. Other programs Compare, Merge, or Generate Updates between two BASIC Programs, check BASIC Sequence Numbers, compare two unsequenced files, and 5 Programs for establishing a Master Directory of several Disks, and sorting, selecting, updating, and printing paginated listings of these files. A BASIC Cross-Reference Program, written in Assembly Language, which provides an X-Ref Listing of the Variables and Reserved Words in TSC BASIC, X BASIC, and PRECOMPILER BASIC Programs.

ALL Utilities include Source (either BASIC or A.L. Source Code).

FLEX, SK-DOS and CCF - \$50.00

BASIC Utilities ONLY for UniFLEX -- \$30.00

MS-DOS to FLEX Transfer Utilities to OS-9 For 68XXX and COOS-9 Systems Now READ - WRITE - DIR - DUMP - EXPLORE FLEX & MS-DOS Disk. These Utilities come with a rich set of options allowing the transfer of text type files from/to FLEX & MS-DOS disks. *CoCo systems require the D.P. Johnson SDISK utilities and OS-9 and two drives of which one must be a "host" floppy.

*CoCo Version: \$69.95 68XXX Version \$99.95

MISCELLANEOUS

TABULA RASA SPREADSHEET from Computer Systems Consultants -- TABULA RASA is similar to DESKTOP/PLAN; provides use of tabular computation schemes used for analysis of business, sales, and economic conditions. Menu-driven; extensive report-generation capabilities. Requires TSC's Extended BASIC.

FLEX, SK-DOS and CCF, UniFLEX - \$50.00, with Source - \$100.00

DYNACALC -- Electronic Spread Sheet for the 6809 and 68000.

*UniFLEX - \$395.00, FLEX, SK-DOS, OS-9 and SPECIAL CCF - \$250.00
OS-9 68K - \$299.00*

FULL SCREEN INVENTORY/MRP from Computer Systems Consultants Use the Full Screen Inventory System/Materials Requirement Planning for maintaining inventories. Keeps item field file in alphabetical order for easier inquiry. Locate and/or print records matching partial or complete item, description, vendor, or attributes; find backorder or below stock levels. Print-outs in item or vendor order. MRP capability for the maintenance and analysis of Hierarchical assemblies of items in the inventory file. Requires TSC's Extended BASIC.

FLEX, SK-DOS and CCF, UniFLEX - \$50.00, with Source - \$100.00

FULL SCREEN MAILING LIST from Computer Systems Consultants -- The Full Screen Mailing List System provides a means of maintaining simple mailing lists. Locate all records matching on partial or complete name, city, state, zip, or attributes for Listings or Labels, etc. Requires TSC's Extended BASIC.

FLEX, SK-DOS and CCF, UniFLEX - \$50.00, with Source - \$100.00

DIET-TRAC Forecaster from S.E. Media -- An X BASIC program that plans a diet in terms of either calories and percentage of carbohydrates, proteins and fats (C P G%) or grams of Carbohydrate, Protein and Fat food exchanges of each of the six basic food groups (vegetable, bread, meat, skim milk, fruit and fat) for a specific individual. Sex, Age, Height, Present Weight, Frame Size, Activity Level and Basal Metabolic Rate for normal individual are taken into account. Ideal weight and sustaining calories for any weight of the above individual are calculated. Provides number of days and daily calendar after weight goal and calorie plan is determined.

FLEX, SK-DOS - \$59.95, UniFLEX - \$89.95

GAMES

RAPIER - 6809 Chess Program from S.E. Media -- Requires FLEX, SK-DOS and Displays on Any Type Terminal. Features: Four levels of play. Swap side. Point scoring system. Two display boards. Change skill level. Solve Checkmate problems in 1-2-3-4 moves. Make move and swap sides. Play white or black. This is one of the strongest CHESS programs running on any microcomputer, estimated USCF Rating 1600+ (better than most 'club' players at higher levels)

FLEX, SK-DOS and CCF - \$79.95

NEW

MS-DOS/FLEX Transfer Utilities For 68XXX and CoCo* OS-9 Systems. Now Read, Write, DIR, Dump and Explore FLEX & MS-DOS Disks. Supplied with a rich set of options to explore and transfer text type files from/to FLEX and MS-DOS disks. *CoCo OS-9 requires SDISK utilities & two floppy drives.

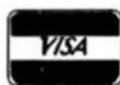
CCO \$69.95 68XXX OS-9 \$99.95

MS-DOS and Macintosh Software at Discounted Prices

"Call for prices, it'll be worth the savings."

(615) 842-4600

Availability Legend
O = OS-9, S = SK-DOS
F = FLEX, U = UniFLEX
CCO = Color Computer OS-9
CC9 = Color Computer FLEX



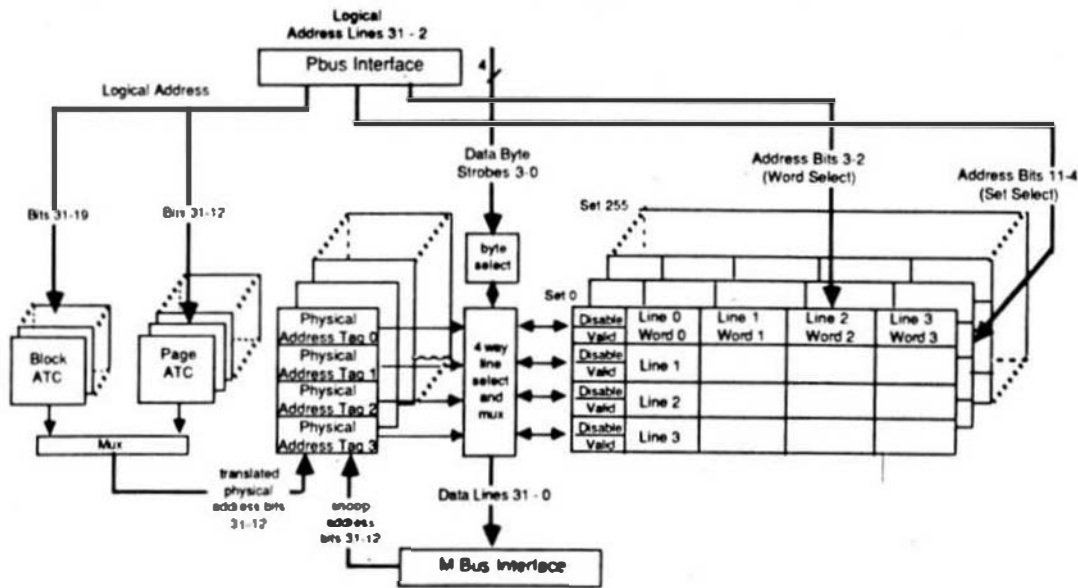
South East Media
5900 Cassandra Smith Rd. - Hixson, In. 37343



**** Shipping ****
Add 2% U.S.A. (min. \$3.95)
Foreign Surface Add 5%
Foreign Airmail Add 10%
Or C.O.D. Shipping Only

*OS-9 is a Trademark of Microware and Motorola. *FLEX and UniFLEX are Trademarks of Technical Systems Consultants. *SK-DOS is a Trademark of Star-K Software Systems Corp.

Figure 6 Cache Organization



supplied to the P-Bus interface to be returned to the processor.

The line structure of the cache, while providing prefetching of code/data, makes best usage of M-Bus burst mode (block transfer) and nibble-mode memories. The number of lines per set also can drastically reduce the probability of replacing a cache line that is going to be accessed in the immediate future.

The cache follows the LRU (Least Recently Used) line replacement rule within a set. It keeps a record of the exact sequential order of line accesses in a set so that the line that is least recently used can be replaced when a cache miss occurs. LRU logic updates the line use order each time a line is accessed or replaced. When a hit occurs, the line that is accessed is marked as most recently used, and the order of the 4 lines in the set is recomputed.

To support fault tolerance, each cache line can be tested and disabled via the disable bit by system software, if found defective.

Cache States And Memory Coherency

The CMMU offers two methods for updating memory; copyback and writethrough. In the writethrough mode, a cache line is written back to memory every time it is modified, keeping memory always consistent with the cache content. The M-Bus traffic can be high when the cache functions in this mode. The copyback policy can be used to maximize memory performance and to reduce bus traffic.

Under this policy, cache writes are written to memory the first time the data in the cache is written (write once); subsequent write to the same location are not written to memory until the cache line is flushed (invalidated or replaced). This delay in updating memory means that the content of memory is not always consistent with that of the cache, making it necessary to have a bus monitoring mechanism that can tell the cache when a bus master tries to access stale global (shared) data in memory. This allows the cache to update memory before allowing other bus masters to try and access the shared data again. The CMMU supports such M-Bus "snooping" capability, to be described in detail in a later section.

The copy back policy requires that

only one cache in a multicache system has a modified copy of any global data. The two valid bits pertaining to a line define the four states of a line as follows:

11 _ Invalid: The line contains no meaningful data

10 _ Shared Unmodified: Other caches may have a copy of this line, and the line is not modified with respect to memory.

00 _ Exclusive unmodified: Only this cache has a copy of this line, and the line is not modified with respect to memory. 01 _ Exclusive modified: Only this cache has a copy of this line, and the line is modified with respect to memory.

The cache control circuitry updates these valid bits on a cycle by cycle basis as the cache lines are being accessed, making the proper state transitions, influenced by the global and writethrough bits pertaining to the data being cached.

M-Bus, the user's interface to the 88000 family

The M-Bus is the user's pathway to the 88000 family devices and the 88000 family's pathway to external physical memory (refer to figure 7). The M-Bus is a 32 bit wide synchronous bus that multiplexes address and data on the same 32 bits and operates at the same clock frequency as the 88200. It is designed to support multiple users and has the ability to function as either a bus master or as a bus slave. The M-Bus includes many features which allow efficient operation in such environments as multiprocessor, multiuser and fault tolerant systems. Prior to discussing these features however, it would benefit the reader to have an understanding of the various signals that make up the M-Bus and their functions. The following is a brief description of the M-Bus signals:

AD0-AD31 (M-Bus Address/Data) form the multiplexed address/data bus. The function of these signals depends on the M bus transaction phase (defined by the M-Bus control signals). During the request phase, the MC88200 drives the physical address onto AD31-AD0. During the data phase, AD31-AD0 are the data input/output lines.

ADP0-ADP3 (M-Bus Address/Data Parity) indicate the parity of the M-Bus address/data lines. The MC88200 uses even parity (parity can be disabled by software), checking parity on reads and generating parity for addresses and memory writes. Each parity signal is associated with one byte of the address/data bus.

C0-C6 (M-Bus Control), when the MC88200 is the bus master, define the transaction phase and type of transaction. The C0 signal defines whether the phase is address (address on AD31-AD2) or data (data on AD31-AD0). During the address phase, C1-C5 indicate intent to modify,

read/write, M-Bus lock, cache inhibit, and global address. During the data phase, C1-C5 indicates end-of-request, read/write, and selected bytes.

These signals are inputs during M-Bus snooping and when the MC88200 is accessed as a slave device.

CP (M-Bus Control Parity), when the MC88200 is the M-Bus master, indicates the even parity of the M-Bus control signals.

ST0-ST3 (M-Bus Local Status) indicate the local M-Bus status when the MC88200 is being accessed as a slave device (register access), or when the MC88200 is snooping a global M-Bus transaction. These signals are inputs during reset and are used to initialize the CMMU ID register.

SS0*-SS3* (M-Bus System Status), are the reply generated by M-Bus slaves in response to the MC88200 address and data phases.

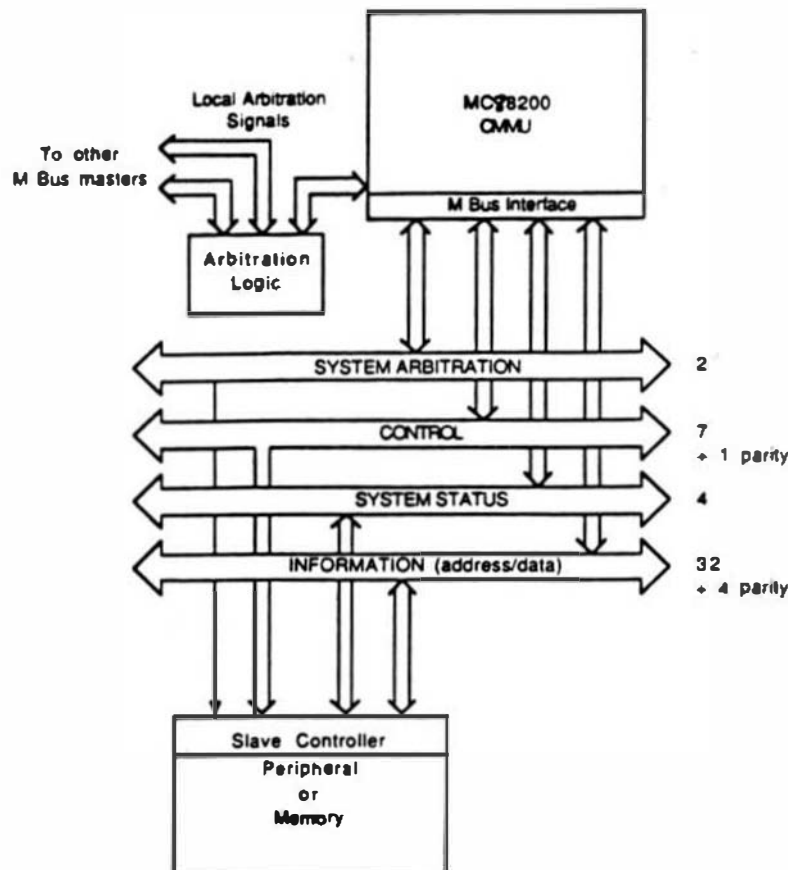


Figure 7 M-Bus Block Diagram

BR (M-Bus Request) is asserted by the MC88200 to request M-Bus ownership.

BG (M-Bus Grant) is generated by external M-Bus arbitration logic in response to a bus request. The MC88200 recognizes this signal only if the M-Bus is not busy (BB* signal negated).

BA (M-Bus Acknowledge) is asserted by the MC88200 when it has received a bus grant in response to a bus request. This signal allows the MC88200 to accept and maintain bus ownership while making memory accesses. It is driven active when BB is negated.

BB* (M-Bus Busy) indicates that some other M-Bus device is currently the bus master, it qualifies the bus grant signal. For an MC88200 to become the bus master, it must receive a bus grant signal from the arbitration logic, and BB* must be negated (no other device is the M-Bus master).

AB* (M-Bus Arbitration Busy) indicates that one or more M-bus devices are performing a bus request. It is an input to the MC88200 that indicates contention for bus ownership is taking place.

MCE (M-Bus Checker Enable) determines the operational mode (master/checker) of the MC88200 M-Bus. In the checker mode of operation, all M-Bus signals are placed in the high-impedance state and all M-Bus outputs are monitored as inputs. The master operates normally. The checker compares its internal results with the results read as inputs. If a mismatch occurs, the checker asserts ERR. ERR can be used to prevent the memory access from corrupting the code, data or i/o system.

MULTIPLE USER OPERATION

Now that the reader has a brief description of the M-Bus signals, it will make it a little easier to follow the discussions concerning the features provided to the user by the M-Bus. The M-Bus operation has four distinct events that take place each cycle; bus arbitration, address selection, data movement and status response. As mentioned earlier, the bus is a multiplexed synchronous bus and all actions on the bus are synchronized to the rising edge of the clock (refer to figure 8). In operations where only one word of information is transferred, it takes two clock cycles to complete the transaction, but transferring information on the bus in 'block mode', only takes $n+1$ clock cycles to complete (in the case of the CMMUs, 'block mode' transfers four words at a time). The CMMUs perform all cache updates and copyback operations in the block mode. Block mode is performed on quad-word boundaries, i.e. the first word transferred starts at address `hex"XXXXXXXX0"` and ends at `hex"XXXXXXXXC"`. At the end of each address and data transfer, including block transfers, all M-Bus participants place their status word onto the bus. This sequence of actions takes place on every

cycle that requires access of data. The following discussion will familiarize the reader with the actions that take place within each major operation on the bus.

BUS ACCESS

M-Bus transactions take place between one bus master and one or more bus slaves. Bus mastership is determined through arbitration while slaves are determined by address decode. The arbitration signals offered on the M-Bus allow the system designer the flexibility to implement a variety of schemes as long as the end result is only one bus master being enabled at a time. The five signals supplied for bus arbitration (described earlier in the signal descriptions) are used to access the M-Bus. These signals work in coordination with each other to provide efficient access and departure on the bus. Below (refer to figure 9) is a simple timing diagram of the interaction of these signals:

It is interesting to note that even though the M-Bus allows only one master on the bus at a time, slave devices are allowed the opportunity at the end of each address or data cycle to respond with status conditions that could potentially cause the current bus master to end its tenure and re-arbitrate for the bus at a later time. This

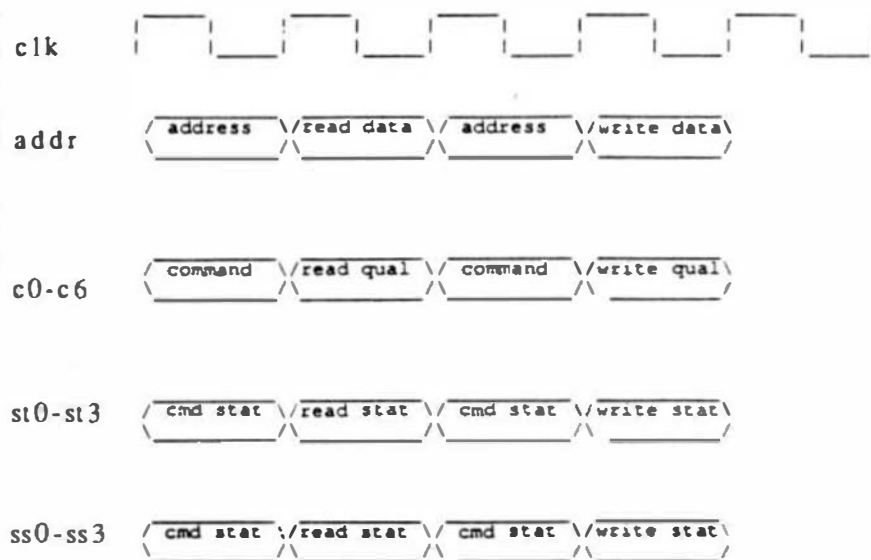


Figure 8 M-Bus General Timing

ability for slaves to impact the bus operation will be discussed a little later. As each potential bus master requires access to the bus they assert BR to the bus arbitration device (implemented externally). AB is the nored output of all BR on the M-Bus (also implemented externally). The arbitration device will then assert BG to the next potential bus master based on its own priority scheme. The device receiving the BG will gain control of the bus as soon as the BB signal is deasserted. BB is deasserted when the current bus master releases its BA. It is possible that during the time gap between receiving BG and BB being deasserted that the bus master elect could lose access to the bus due to a higher priority device being given BG and their BG being removed or that a bus cycle ended in a status of retry or error. In either case the master elect would have to re-arbitrate for the bus. Once a device does receive ownership to the bus they will assert BA. The bus master may retain the bus until other devices require access either for new transactions or to recover from exception conditions. Once the current bus master loses the bus, there is an option in the CMMU that will allow that device to immediately reassert BR (Priority Request) if necessary or wait until AB is deasserted (Fairness protocol) before the BR is asserted.

ADDRESS GENERATION AND DETECTION

The request (address) phase of a bus access is indicated by control bit C0 being asserted. At the end of the request phase, C0 is deasserted and the bus operation enters the data phase. In the master mode of operation, addresses are normally output to select a slave device. In certain conditions however, the address generated may contain the control register address and ID of one of its own internal registers. This is called a PIRA (P-Bus Internal Register Access). In this particular condition, the CMMU acts as both a master and a slave. It acts as a bus master by arbitrating for the bus and generating the address; and acts like a slave by decoding the address. On each single word access, the address is output and a reply is received to either validate that address or to indicate an exception condition. In block transfer

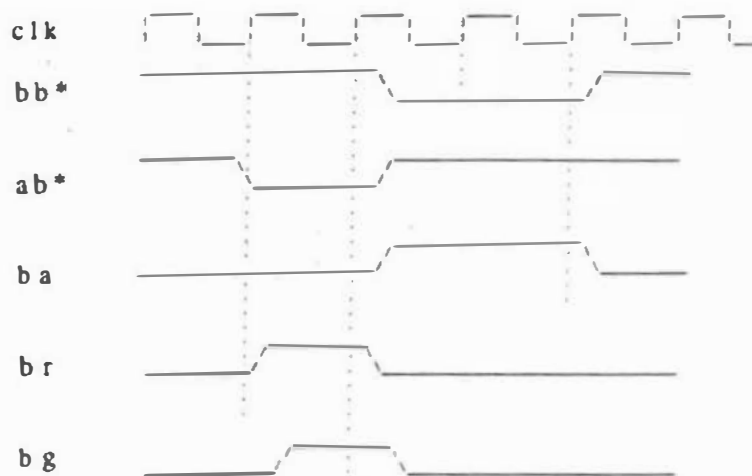


Figure 9 Bus Arbitration Timing

mode the address is generated for the first cycle but not for the last three cycles. It is the responsibility of both the master and the slave to update their respective address counters internally while performing block transfers. The CMMU, while functioning as a slave device, does not use a chip select in the normal sense, but has an ID which is loaded into the CMMU at power-up and can be loaded by software as well. This ID is checked on each request phase to determine if that access is intended for that particular CMMU. The ID has to accompany every address that is intended to access a CMMU from the M-Bus or the P-bus. The request phase is terminated when the OK reply is received via SS0-SS3.

READ AND WRITE ACCESS

Data transfers across the M-Bus are accomplished via reads or writes in either a single or a block mode. In single word transfers the data is preceded by an address and followed by the appropriate status to confirm the operation. If the user intends on transferring less than one word of data, a single word transfer is executed with the appropriate byte strobes enabled. In block mode, the data is transferred in burst of up to four words. A request phase precedes the first word transfer with an address that is byte aligned. Each word transfer is followed by a status condition that qualifies that word. Block transfers are terminated by the master asserting an End of Request (control pin C1) in the data phase of the last word or by the slave asserting an End of Data status for the last word it can handle via SS0-SS3.

STATUS REPLIES

One of the very unique features of the M-Bus operation is the status reply phase. In this phase, all operations are validated, invalidated or redirected depending on the response to that phase by each participating slave device. These status replies are communicated on the bus via the system status lines described earlier. The information conveyed by this status mechanism is how the M-Bus is able to support such features as multiprocessing. The status lines are mutually exclusive with an inherent priority defined through the use of active low level inputs. The status reply with the most active low inputs has highest priority.

Every request and data phase has a corresponding status response presented on the M-Bus at the end of that phase. Since only one master is active at a time, it is the responsibility of each active slave to place the appropriate status on the bus at the end of every phase (refer to figure 8). There are five possible responses that can be placed on the bus by participating slaves: 1) Error 2) Retry 3) Wait 4) End of data 5) OK.

The Error response is normally placed on the bus by a slave device that has detected a parity error, but system implementation allows the developer of M-Bus slaves to apply other error conditions to this response. When an error is detected, the bus master loses ownership of the bus and has to return to the bus arbitration phase.

The next status response, Retry, is the normal mechanism used in multiprocessor systems to handle Bus Snooping (Snooping is described in more detail later). It is also possible for Retry to be used in certain applications as a priority mechanism for higher priority devices to gain access to the bus as soon as it is required.

The Wait response is used to delay the progress of either a request or data phase for various systems reasons. It can be applied by memories with slow access times or by memory controllers who are capable of setting up blocks of memory for burst operations. CMMUs that are Snooping will assert Wait states to do address interrogation. As long as Wait states are asserted by any slave device, the current cycle, whether it is a request or data phase, is repeated on the bus.

End of data (EOD) is used by participating slave devices in the data phase to terminate a burst transfer if the burst cannot be successfully completed. It will also be asserted by CMMUs if a burst is attempted by another device and that burst is not quad-word aligned. In the request phase EOD is decoded as a proper OK response. EOD also allows a slave to respond with a single word when a CMMU requests four words. The CMMU will then request the next three words one at a time.

OK is the normal response to operations that complete successfully, whether they are single word transactions or burst transactions.

BUS SNOOPING

Bus snooping is the unique feature supported by the 88200s to facilitate multiprocessing through the use of shared (global) memory. For those not familiar with the term Bus Snooping, it is the ability of caching devices to monitor the memory accesses of current bus masters and to intervene if the memory location the master is reading or writing is shared memory and has been modified by that caching device, but not updated in memory. When a slave device asserts Retry in response to a memory access, the bus

master will immediately relinquish the bus and wait one complete cycle before re-arbitrating for the bus. The one cycle of dead time is to allow the device who asserted Retry to gain control of the bus and update memory. In order for this snooping function to occur, snooping has to be enabled in the CMMU and the memory location being snooped has to be flagged as global. Snooping CMMUs monitor only those bus cycles that are accompanied by the global bit (C5) being set. Each request phase is given an Ok reply by snooping CMMUs. However, the data phase of a global transaction is given two Wait states to afford the snooping CMMUs time to determine if there is a match (snoop hit) in their cache tag. If no match is made, the snooping CMMUs will respond to the next data phase with an OK. If a match is detected and that location is dirty (contains data that is different from that in memory), the detecting snooper will assert Retry on the bus and then BR. The next cycle will be given to the snooper who in turn will update memory with the correct information. If the snooped transaction is a CPU read, the data is copied back to memory, but retained in the cache of the snooping CMMU with different status. If the snooped transaction was a CPU write, the data is copied back to memory and removed from the cache of the snooping CMMU. Snooping is absolutely essential in multiprocessing systems that use the "write back" policy for their respective CMMUs.

FAULT TOLERANT SYSTEMS

There are several features supported on the M-Bus that will benefit fault tolerant systems designs. Features such as output pin comparison, shadowing and parity generation and detection help the system implementer to achieve a more reliable system at a lower cost.

Every output pin on the M-Bus has an internal comparator that compares the output value on the actual pin to the input

value to the output driver. Any mismatch will drive the ERR pin active indicating a bus fault.

Shadowing is implemented on the M-Bus by setting a specific input pin (MCE) on the CMMU. When MCE is set, all M-Bus signals are placed in the high impedance state and all outputs are monitored as inputs. The shadowing CMMU monitors all of the transactions from the master and if any mismatch occurs, the ERR pin is asserted.

Parity generation and detection is enabled by setting a control bit in the System Control Register of the CMMU. When enabled, even parity is checked for all read data phases as well as for the control signals that accompany each cycle. Parity is generated on all request phases and write data phases along with parity for the outgoing control lines C0-C6. Any parity errors detected on the M-Bus are reported back to the processor via the System Status Register.

The CMMU also has an internal feature that allows the user to test each line of the cache and to disable any cache lines that test faulty. This provides the user with a mechanism that does not require the storing of a fault map for the cache that has to be referenced on each access.

As can be seen by this brief look into the features and operation of the 88200 and associated M-Bus, Motorola has created a very powerful, yet straight forward approach to handling the many systems concerns that normally surround the design of performance oriented applications. For a more detailed understanding of the features and functions described in this article, Motorola has available Technical Summaries and User's Manuals for the 88100 and 88200 as well as working devices and all necessary software such as compilers, assemblers and functional simulators.

FOR THOSE WHO NEED TO KNOW

**68 MICRO
JOURNAL™**



The Macintosh® Section

Reserved as

A place for your thoughts

And ours.....

Mac-Watch

OF MICE AND MACS . . .

Reviews of Stepping Out II and Quickeys; Information on an upgrade to ImageStudio

*By James E. Law
1806 Rock Bluff Rd.
Hudson TN 37343*

How Many Mice?

It's always interesting to check out the Macintosh software reviewed in other magazines, especially if it is for a product I have reviewed or which I use. I sometimes find myself saying, "How can they give 5 'mice' (out of a maximum possible of 5) to that dog of a program!" On other occasions it's "That guy is obviously prejudiced against the software. He's picky!" The truth of the matter is that the relationship between a user and his software is a very personal thing and there is room for honest differences of opinion.

Speaking of differences of opinion, why does nearly every program reviewed by some magazines rate a 'superior' rating. Of the 24 programs listed on a single page of ratings in a certain Macintosh magazine, 22 were rated 4 mice or better and 6 programs were rated 5. Is the current batch of Macintosh software that good, or do we just have low expectations? Surely it doesn't have anything to do with advertising dollars? I'm glad to say that the management of Micro 68 Journal has never tried to influence the conclusion of my reviews, even in the case of those few that were quite negative.

A Review of Stepping Out II

The advertising hype for Stepping Out usually says that it appeases one's desire for a large monitor. If so, it would be well worth its modest price. We all get tired of looking at the Mac's little 9" monitor. (I just don't get so tired of it that I want to spend \$1500 for a large-screen monitor.) The truth is that Stepping Out's function is to allow you to easily move around a document that is larger than the screen. As the mouse is moved so that the cursor approaches the edge of the screen, the screen image scrolls smoothly to expose more of the document. A secondary function of Stepping Out is to provide for reduced images of your document for page previews and magnified views for detail work. Both views are interactive, that is, you may perform work while the image is reduced or magnified.

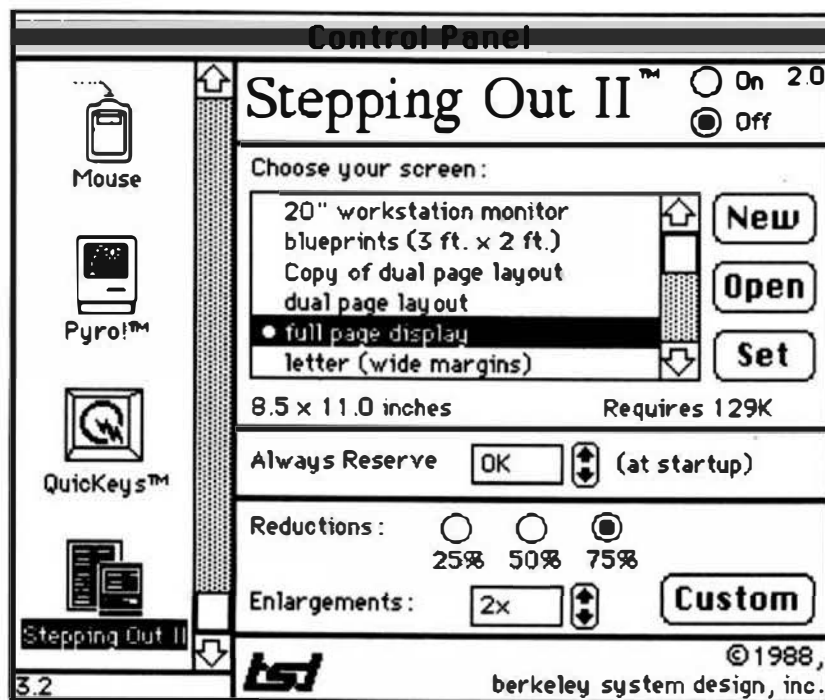
The first version of Stepping Out was generally accepted as being a masterful bit of coding, but some users complained about jerky scrolling and slowed operations. The primary purpose of the latest revision, Stepping Out II, was to improve performance in these areas. To a very large extent, this effort by Berkeley System Design, Inc., was successful. Stepping Out II scrolls quickly and smoothly. While there is still a noticeable reduction in screen refresh time, it was not enough to be objectionable.

Stepping Out II is an INIT, that is, it must be loaded in the system folder and then, upon restarting your Mac, it is ready to use. If you open the control panel you can then turn Stepping Out on and choose the desired parameters. You may choose a variety of document sizes from full size display (i.e., 8-1/2 x 11) to blueprints (i.e., 3 ft x 2 ft). If none of the pre-entered sizes suit you, you may set up a custom size. The control panel also allows you to set parameters for reductions and enlargements and to specify the amount of memory

to be reserved upon start up for Stepping Out II.

Stepping Out II then constructs an "image" in memory of the size page you specified. As the cursor approaches the edge of the monitor, the 'preconstructed' image is fed to the screen so as to provide the perception of smoothly scrolling to show more of your document.

I tried Stepping Out II with page layout programs (PageMaker, ReadySetGo), word processors (MacWrite, WordPerfect), desktop presentation programs (ReadySetShow, PowerPoint), and graphics programs (SuperPaint, Canvas), and very few incompatibilities were identified. The only problem noted was that PowerPoint windows refused to scroll properly under Stepping Out II. Of course, programs with non expandable windows like MacPaint and MacWrite gain little from the use of Stepping Out II (although you can quickly scroll up and down the page in MacWrite).



Stepping Out II Control Panel

If you specify a very large document size, it can be frustrating to work in one corner while the memo bars and tool boxes are at the other corner. Stepping Out II allows you to lock memo bars and tool boxes into place so that they are always visible, no matter where you scroll in the document. The feature does not appear to work with Canvas hierarchical menus that pop out into the area reserved for the document itself.

Stepping Out II makes it easy to magnify portions of the screen up to 16 times its normal size for close up work. The size of the area to be magnified and the degree of magnification is ad-

justable. This feature works well. The reduction feature does not work so well. Only 25%, 50%, and 75% reductions are allowed. The resulting screens are quite unattractive as the desktop background turns black.

Stepping Out II works with any MacPlus, SE, or Mac II. It is advertised to work with all standard and large Mac II monitors except for 24 bit color. It does not work with many of the big screens used with the MacPlus and SE. Stepping Out is designed to be compatible with all software that works with large screen hardware.

Stepping Out II seems to work, as advertised, with the few problems mentioned above. The question is whether you need extra help in moving around your documents. If scroll bars and built in 'pusher hands' aren't good enough for you, perhaps you need Stepping Out II.

Back to the business of assigning overall ratings. . . Five mice is a lot of mice. I would give Stepping Out II, 4 mice, a solid value; but with room for improvement.

How Quick is Quickeys?

Now I don't want to give up my mouse, but sometimes it's a pain. In the middle of a series of keyboard actions, I have to stop, use the mouse, the return to the keyboard. In the Macintosh Plus manual, there is a picture of a Mac user sitting way back in his chair away from the screen with keyboard in his lap and feet on the desk. We all know that in the real world, this pose wouldn't last for over a few minutes before he would have to reposition himself to operate the mouse. Wouldn't it be nice if some of those mouse operations could be handled from the keyboard?

Well, they can with Quickeys

from CE Software. Mouse actions such as choosing menu items, clicking dialog boxes and scrolling the screen can be assigned to a key or combination of keys on the keyboard. A single keyboard stroke can also be programmed to perform other actions such as entering a passage of text.

Quickeys, is an INIT and is activated by placing it in your system folder and restarting your Mac. You then, at any time, have access to the Quickeys control box by typing OPTION-CONTROL-RETURN. The combination of keys, like all other Quickey preestablished by key combinations, may be easily changed to suit the individual user.

Quickkey entries are made by selecting one of 11 options under the "Define" menu. A very brief overview of the possibilities are as follows:

1. TEXT — Assign a string of text to be displayed when a key is typed.

2. FILE — Close the currently open file and open another application or document without going through Finder.

3. MENU/DA — Open any DA or make any menu selection with a key stroke.

4. ALIAS — The typing of a specific key will cause a different character to be displayed. Use this feature to reorganize your keyboard.

5. CLICKS — Use a keyboard command to replace any mouse click and dragging operation.

6. SEQUENCES — String up to 30 Quickkeys entries together into a powerful macro.

7. Buttons—Use a keyboard command to click any button.

8. MOUSIES — Use keyboard commands for specific actions such as zoom or close window, page up or down, or line up or down.

9. SPECIALS — Use keyboard commands for miscellaneous actions such as shut down, re-start, or select rear window.

10. DATE/TIME — The designated key combination makes the current date and/or time appear in one of a number of formats.

11. F KEYS — Select an FKEY with the designated key command.

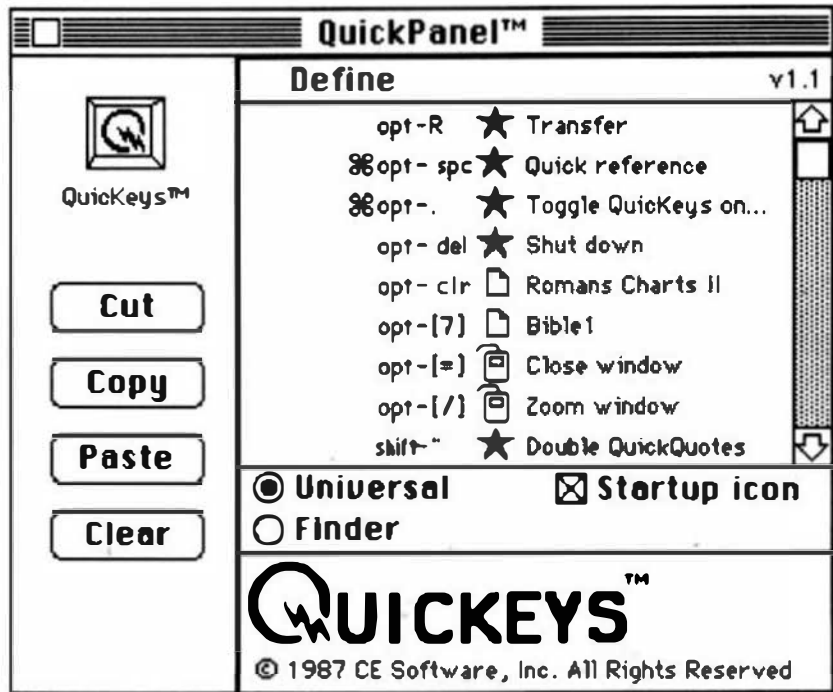
Each Quickkey key stroke can be assigned to a "universal set" (it is always available for all applications) or may be assigned to a specific application. In cases of conflicts, the key strokes in the program specific sets take precedence over those in the universal set.

To illustrate the usefulness of Quickkeys, let me tell you how I have made use of it to speed up my preparation of viewgraphs with PowerPoint. First, I press OPTION-CLEAR to go directly to PowerPoint without going through the Finder. OPTION and "+" or "-" lets me flip through my slides from the keyboard. With OPTION-Z, I zoom back to review the entire slide or with OPTION-A I can expand the slide to full size. OPTION + (arrow key) allows me to quickly page up or down, right or left. OPTION-COMMAND - (arrow key) lets me move more precisely a line at a time. Finally, when I press OPTION-RETURN, my Mac immediately shuts down (after giving me a chance to save) without

going through the Finder.

Since I assign the key strokes, I can select something that makes sense to me. It's easier to remember that way. But if I forget, I can enter OPTION-COMMAND-SPACE BAR and a list of all my Quickkey entries are displayed.

Quickkeys is a handy program. Even with the standard Macintosh keyboard, it is a real time saver. Power users and experienced typists will appreciate not having to take their hands off the



Quickkeys Control Panel

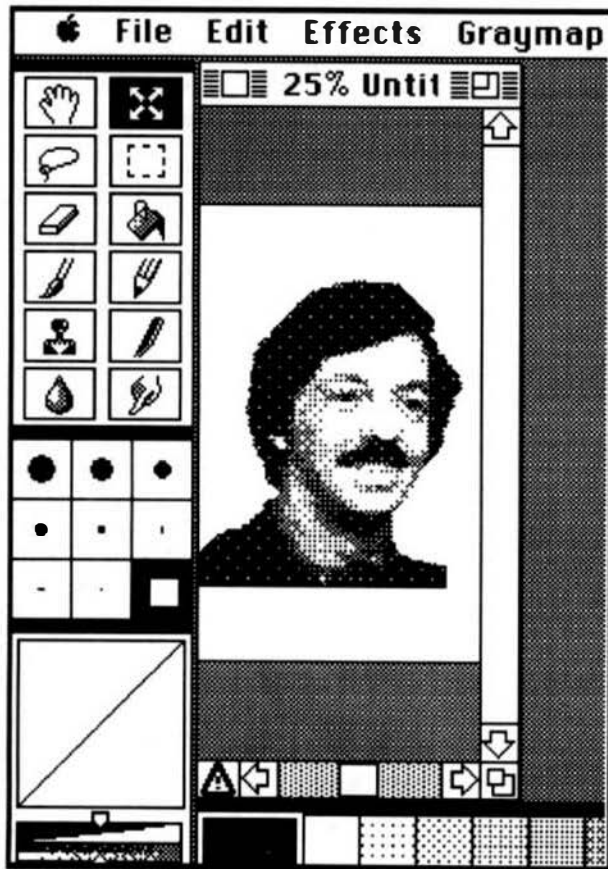
keyboard to reach for the mouse. I would have to rate it at least 4-1/2 mice if I were giving ratings based on mice points.

Image Studio is Upgraded

Version 1.0 of Image Studio software from Letrasel USA, was named as product of the year by Personal Publishing magazine and now an ever more powerful version (1.5) is available. Image Studio allows you to produce high quality camera-ready halftones on the Macintosh and incorporate them into desktop published documents. The principle improvements in version 1.5 are as follows.

1. Enhanced Grayscale Capability — ImageStudio 1.0 could process up to 64 shades of grey but with Version 1.5, you can manipulate 256 shades. Obviously the number of shades in a file depends on the capabilities of the scanner used. But once you have the image in ImageStudio, you can take advantage of all 256 shades to fine tune the image.

If you have a Macintosh II with an 8-bit video driver and a high resolution monitor, you can view 102 shades on screen. With the Macintosh Plus and SE, you can view 64 shades plus white on screen.



Partial ImageStudio 1.5 Window

2. Memory Management — Version 1.5 provides far more sophisticated memory management techniques which make it possible to work with images larger than your RAM would normally allow. This is important because scanning with 256 shades and 300 dpi resolution creates very large files. ImageStudio handles this feat by creating a working copy of the image on disk. You can then open a portion of an image at a time while you edit.

3. Scanner Control — You can now drive scanners and image grabbers without leaving ImageStudio. The "File" menu now includes

"scanner setup" to set options and a "scan" command which activates the scanner or image grabber. This arrangement allows you to scan directly into RIFF format which has the same resolution potential as TIFF, but with smaller file sizes.

4. Upgrade of Tools — Finally, a number of ImageStudio's tools and commands have been enhanced. For example, you can now scroll selections past the window boundaries and can constrain selections to squares or straight lines.

Version 1.5 of ImageStudio greatly enhances an already powerful program. Those of you who have occasion to use scanned images should check it out.

Please Feed Me Back Some Feedback

I would like to hear from you on what you would like to see in this department of 68 Micro Journal. Would you rather see a number of shorter reviews or a few more in depth reviews? Is there any particular type of software you would like to read about? Let me hear from you.

EOF

FOR THOSE WHO NEED TO KNOW

**68 MICRO
JOURNAL™**

Writing Position Independent and Reentrant Code for the MC68000 Family



MOTOROLA INC.

3501 Ed Bluestein Boulevard
P.O. Box 8000, Austin, Texas 78762

By Truman T. Van Sickle

Background

A multitasking system is a computer operating system that will allow the seeming concurrent execution of several distinct tasks within a single computer. The several tasks might be completely independent, or mutually dependent in the sense that calculations or measurements made by one task are needed as inputs to another. For example in an industrial control system, one task could be used to set several digital and analog outputs to control a machine. A separate but communicating task could monitor sense switches and analog signals from the machine. The computer inputs indicate the state of the machine at any time and dictate the next operation that the machine must execute. The machine state might be stored in an area of memory that is accessible by either task. With these two tasks running concurrently, the programmer will have a table with the current state of the machine. It is not needed to scan the input status as is done with programmable controllers or single tasking machines to determine the current machine status. The job of the programmer is

easier in this case with a multitasking system than with a single tasking system.

Some computers employ memory management. A memory manager is a hardware feature that examines the logical address of each computer transaction and determines the correct physical address for this activity. When a programmer writes a program, it is usually started at an address zero. When this program is executed, this beginning address must be changed if for no other reason than the MC68000 family of parts dedicate the first 1024 bytes of memory to an exception vector table. When the computer has no memory manager, it is required that the programmer allocate memory for the various memory resident tasks. In this case, it might be necessary to place the code to be executed anywhere in memory where there is sufficient unallocated memory to hold the task. Code that can execute when placed anywhere in memory is called position independent.

Reentrant code is a feature that will allow more than one task to appear to execute a routine at the same time. The concern with reentrant code is that the machine and memory status associated with the code be secure enough that the routine can be arbitrarily stopped at any point of its execution and another task can start executing the same code with no interference between the tasks calling the routine.

Features like reentrancy and position independence were not really considered necessary for microprocessors until the Motorola MC68000 family of parts became available. The main need for reentrancy comes about when multitasking operating systems are used. Position independence is desirable for many programs in a multitasking system, especially a multitasking system with no hardware memory management. The purpose of this paper is to describe the need for both position independent and reentrant code, and to demonstrate how easily these important features are implemented with the MC68000 family architecture.

Reentrant Code

Reentrant code means that several different tasks can call a reentrant module, seemingly at the same time, without interference. Suppose that a multitasking system is operating, and at the current instant, a specific routine is being executed. Let us use a program like a software-floating point multiply as an example. Task A is in the process of computing the product of two numbers. At any time, a clock interrupt can occur and remove control of the processor from Task A to the task dispatcher of the operating system. When such an operation occurs, the task dispatcher can, and usually will, change the running task from Task A to another task that is on the ready list. Let us assume that Task B is placed in control of the processor. This process involves first saving the status of the computer for Task A and then restoring the previously saved status of the machine at the last time Task B was in control. Task B is now running.

To make the problem more interesting, let us suppose that the software floating point multiply is a globally shareable routine. Globally shareable means that anytime a software multiply is needed, the same routine will be used by any task. Only one copy of the multiply routine will be stored in memory. Everything is okay until Task B needs to execute a software multiply. At this time, a routine must be entered that was being executed by Task A when Task B took control of the processor. There can be no special provisions at the exit of Task A control to save the status of the routine being executed. The normal exit procedure at task switch is to save the machine status. Therefore, the status of the executing reentrant routine must be contained completely within the machine status at all times.

For the most part, the machine status contains the program status. The Data and Address Registers are all saved as part of the machine status as is the content of the Condition Code Register and the Program Counter. The only missing components of the program status are the values contained in data storage needed for the execution of the routine.

A simple way to avoid storage problems is to store all data in the memory space of the calling program rather than to have specified data space in the executing routine. These data are all stored on the stack of the calling program, and when the stack pointer is saved as part of the normal machine status, all volatile data associated with the executing routine is automatically saved in

memory associated with the calling program. When Task B starts to execute the floating point multiply, any necessary volatile memory will be assigned in the Task B memory space, and the data saved while Task A was being executed will be safely saved in the memory space of Task A. This operation can be repeated indefinitely and the number of tasks that can simultaneously access a given reentrant routine is unlimited.

Another application of reentrant code is to implement recursion in a routine. A function is said to be recursive when it calls itself. If a routine is reentrant, it makes no difference whether the routine calls itself or it is reentered by another program after a task switch. In either case the handling of local storage must allow the routine to be entered while it is set up and executing a call from another program. Any recursive routine must be reentrant.

Suppose that a multiuser operating system is being used by several programmers. Much of the time that programmers sit at the keyboard they are either editing or compiling or assembling a program. With a well designed operating system and properly designed system utilities such as editors and compilers, there is no need for multiple copies of these routines to be resident in memory. Assemblers, editors, compilers, and other frequently used programs are each made reentrant. The program loader can examine the task list when a new program is loaded and make use of a copy of the program that already resides in memory. This approach can

provide significant memory savings on a large multiuser system.

Writing Reentrant Code

Most programming is done in high level languages. Usually the language takes care of providing reentrant code. All C compilers and most Pascal compilers can provide reentrant code. These compilers create reentrant code so long as there are no static variables associated with the routines. All variables should be dynamic. Dynamic variables are stored on the stack which is passed from the calling program. Static variables must be stored locally and associated uniquely with the local routine. Therefore, static variables will not be saved automatically when the task dispatcher passes control from one task to another, and there is no guarantee that a second task can enter the routine without disturbing the status of the first.

Static variables can be used if they are not used for execution of the routine. For example, a routine might require initialization the first time it is entered. Here, a static variable could be used as a flag to indicate that the initialization has been completed. This variable is not a part of the computation executed by the routine whenever it is called. Generally, static variables must be used with extreme care with reentrant code.

The discussion here is aimed at assembly language programming only. Most of the time, high level languages will suffice for the every day programming. The times that the high level languages fail

are those when use of and access to machine level operations are necessary. For example, I/O drivers, screen access programs, graphic drivers, and global functions such as floating point operations would probably be written in assembly language. Both reentrancy and position-independence are required for these types of routines.

The MC68000 family parts have architectural structures and instructions that make the programming of reentrant code particularly easy. The indirect addressing modes allows addressing of bytes, words or long words — 8 bit, 16 bit, or 32 bit quantities — that are offset from a frame pointer or from the stack pointer.

A most convenient instruction is the LINK instruction. The format of the link instruction is

LINK A_n, #displacement

In response to the link instruction, the MC68000 pushes the content of the designated address register onto the stack. Then the new value in the stack pointer after the push operation is placed into the designated address register. Finally, the value displacement is added to the stack pointer. In the MC68000 family, the stack pointer is always the address register A7. After execution of the LINK operation, there is a designated amount of memory on the user stack that can be used as storage by the currently executing routine. Boundaries of the storage are marked by the contents of the stack pointer and the contents of A_n.

The diagrams in Figure 1 shows the impact of the LINK instruction. In this case, the designated address register is A6, and the displacement is -14. The memory entries shown in Figure 1 are 16 bit words. Assume that the program has

UNLK An

This instruction causes the contents of An to be placed in the stack pointer, and the top of the stack is pulled into the register An. After this operation is completed,

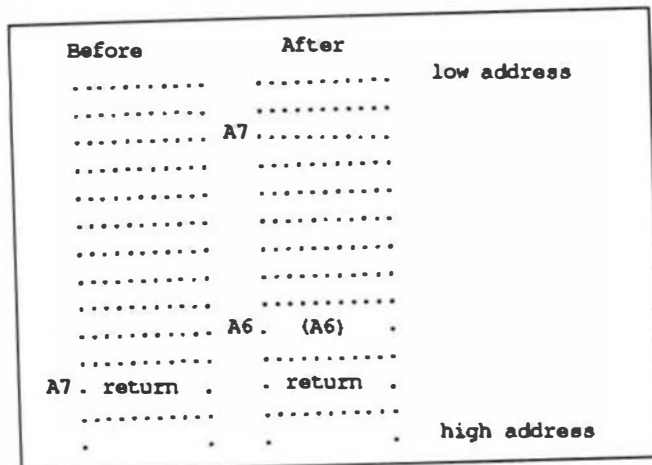


Figure 1. Effect of execution of the instruction LINK A6,#-14

just entered a subroutine. Therefore, the stack pointer is pointing at a location in memory that contains the long word address of the next instruction in the calling program. This location is designated by return in the figure. After the LINK instruction, the content of the register A6 has been pushed onto the stack, and the new value of the stack pointer has been placed in the register A6. Finally, the displacement -14 has been added to the content of A7 so that the final memory configuration is as shown in the figure.

At the end of the subroutine execution, the original status is restored by execution of the instruction

the memory status is restored to precisely that shown in the Before portion of Figure 1.

After execution of the LINK instruction shown in Figure 1, there are 14 bytes of memory allocated to the routine. These 14 bytes will remain attached to the routine until an UNLK instruction is executed prior to return to the calling program. After execution of the UNLK, the memory assigned to the routine is gone. Thus, the name dynamic. The amount of storage assigned to each routine is just the amount needed to execute the routine, and the storage exists only while the routine is being executed. Also, it should be noted that in the event that the routine is interrupted by the task dispatcher, all registers will be saved prior to the starting up of another task. Therefore, when this routine is restarted by the task dispatcher at a later time,

all registers will be restored and any dynamic memory used by this routine will be unaltered in the event that another task might use this same routine.

The register An used with the LINK instruction is called the frame pointer. Mnemonic names may be assigned to the memory locations on the stack by any of several means. Probably the easiest technique is to use the OFFSET directive in an assembly language program. An

Here the table is terminated by entry into a SECTION portion of the code. When this table is assembled, the value for ABLE will be 0 because the offset is 0. BAKER and CHARLIE will be 4 and 8 respectively. DOG will be assigned a value of 10. Reference to these mnemonics relative to the stack pointer will automatically access the correct memory location on the stack. An example that accesses these locations is as follows:

SUB LINK A6, #-14	Link the stack space
MOVE.L D0, ABLE (A7)	Place D0 into ABLE
MOVE.W D0, CHARLIE (A7)	Place the lower word of D0 in CHARLIE
SUB.L D0, D1	Subtract D0 contents from those of D1
MOVE.L D1, DOG (A7)	And save the result in DOG
etc.	
UNLK A6	Unlink the stack space
RTS	

OFFSET directive allows definition of a table of offsets created by Define Storage directives. Symbols thus defined are kept internally by the assembler. The offset table may contain no executable instructions. The offset table is terminated by a SECTION or ORG directive.

As an example, suppose that our routine needs three long word variables, ABLE, BAKER, and DOG, and an integer CHARLIE. The following OFFSET table could be used to create these variables:

	OFFSET 0
ABLE	DS.L 1
BAKER	DS.L 1
CHARLIE	DS.W 1
DOG	DS.L 1

SECTION 1

Here all memory operations are relative to the stack pointer. Figure 2 shows how the memory is organized on the stack.

Sometimes the stack is used for other operations during the execution of the routine. In such a case, the stack pointer might be altered during the execution of the program, and there references into the offset table would be shifted. Here all of the memory values would be lost unless extreme care were executed in creating the code. An alternative means exists that allows memory references relative to the FRAME POINTER which should remain unaltered during execution of the routine. Note the following minor change in the OFFSET table.

OFFSET -14

ABLE	DS.L	1
BAKER	DS.L	1
CHARLIE	DS.W	1
DOG	DS.L	1

SECTION 1

This symbol table starts with an offset of -14 rather than 0 as was used earlier. Here, ABLE will be assigned a value of -14, BAKER a value of -10, CHARLIE will be -6 and DOG will equal -4. Recall that A6 is pointing to the top of the frame pointer. Therefore, offsets generated by this table relative to A6 will choose memory locations for the various labels within the stack area attached by the link command. The listing below performs the same functions as the example program above.

```
SUB LINK A6, #-14
  MOVE.L D0, ABLE(A6)
  MOVE.W D0, CHARLIE(A6)
  SUB D0, D1
  MOVE.L D1, DOG(A6)

  Etc.
  *
  *
  UNLK A6

  RTS
```

The difference in this case is that all labels are offset from the frame pointer A6 rather than the stack pointer A7. In most cases, either approach works equally well and the choice of which reference technique depends only on the programmer's whim.

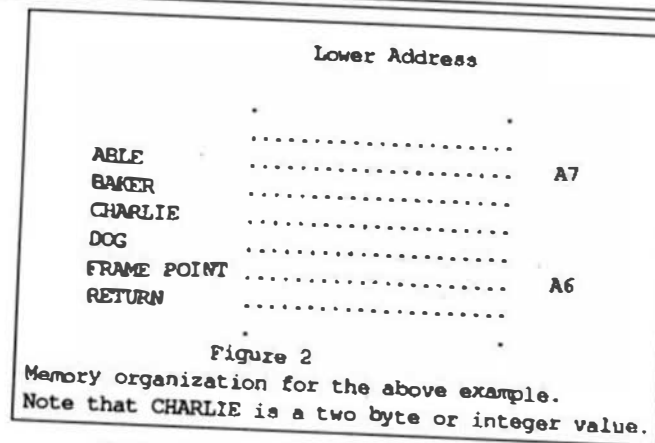


Figure 2
Memory organization for the above example.
Note that CHARLIE is a two byte or integer value.

POSITION INDEPENDENCE

Position independent code can be placed anywhere in memory and will work with no difficulties. There are several reasons to want position independent code. Most code intended to be placed in ROM should be position independent. Therefore, the ROM beginning address need not be kept at a specific memory location. Another, and probably more compelling reason, has to do with memory assignment on multitasking systems where there is no memory manager. If a system has no memory manager, there are two frequently used approaches to the assigning of memory space to a program. The first requires that the programmer link the several programs to be loaded at unique addresses so that none of the programs can overlap. This approach really makes the programmer into a makeshift memory manager. When it is required to load this code, the loader must not allow the memory space of this program to overlap the memory space of any other running program. A somewhat better approach requires that all code be position independent. When a position independent module is loaded, the loader must

merely find free memory space large enough to contain the program and load it there. Once the program is started, it will operate exactly the same as if it had been assigned to a designated memory space. Since the system is multitasking, there will be several modules loaded into the memory space at a time, and the operating system has the responsibility of making certain that there is no memory interference between the modules. Otherwise this job is faced at link time by a programmer who might not even know what modules will be loaded into the computer at any given time.

Writing Position Independent Code

A provision in many assemblers allows grouping blocks of code or data into sections. Motorola assemblers for the MC68000 family of parts provide for sixteen sections for each module. Section organizations allow the programmer to specify special characteristics that need to be grouped when the program is linked into a working module. For example, dynamic ram storage could be placed on one section and high speed

staticram in another. Utility programs stored in ROM could be in a section, and applications programs could be linked into yet another section.

When a program is assembled, a symbol table is created that contains the relative address of every label used in the program. These addresses are relative to Section starts with the MC68000 family assemblers. If there are several sections within a given program, each section start address is set to zero. Therefore, the symbol table must contain both the relative section address of each label as well as the section in which it is contained. Code generated by an assembler is of the relocatable object format.

Another parameter kept by the assembler is the current location counter. This value can be accessed by the programmer by use of an asterisk in the operand field. For example, a command to load the current location would be

```
LEA *,A0
```

This instruction will cause the current value of the location counter to be loaded into the address register A0.

One or more relocatable object modules are combined into a single loadable object module by the linker. The linker pulls all of the like sections together and assigns them to memory locations. After linking, the program instruction addresses, earlier called location counter values, will be called program counter values. When a program is loaded into memory and the program counter

is given the value of the first instruction in the program, the program begins to execute. The instruction pointed to by the program counter is fetched into the machine to be executed, and the program counter value for the next instruction is calculated before completion of the first instruction. This sequence is repeated progressively until the program is completed.

It is not necessary to place the various sections of a program in contiguous memory. In fact, if the sections are used as outlined above to designate special memory characteristics, it would be impractical to always expect the data from any program to occupy contiguous memory. Therefore, the location of data storage in a computer may not ever be position independent.

The above discussion might lead us to believe that the program, after linking, must be placed at some designated location in memory and remain there. Seemingly, data addresses must be fixed in some way so that they can be found when different portions of the executing program require them. There is a way around the problem. MC68000 instructions can use the difference of the data address and the program counter to locate a data address. Such an addressing mode is called Program Counter Relative, PCR. If all data accesses and branches — including branches to subroutines — are PCR, there is no need for absolute addresses anywhere in the program. If no absolute addresses are needed, then the

program can be executed anywhere in memory, and it is indeed position independent.

To create position independent code, the programmer must make all data accesses and changes in program flow, JMP, JSR, BRA, Bcc, and BSR program counter relative. Any branch instruction is automatically PCR. The only concern with a change in program flow is to make certain that the branch instruction can reach the destination. This concern will be discussed at greater length later.

For data accesses, the PCR mode can be used. Function codes of the MC68000 parts contain hardware signals of a data access or a program access. Any storage operation that is PCR implies that the operation is both program and data at the same time. A PCR operation causes access of the program counter which will assert Program Function Code line. The fact that anything is being stored in memory will cause the Data Function Code line to be asserted. These two lines cannot be asserted at the same time. Therefore, this type of operation is not allowed in any of the MC68000 family of parts. The addressing category that designates an effective address in which the data can be stored is called Data Alterable. Data Alterable destination effective addresses for a PCR operation are specifically not allowed. Suddenly, the problem becomes a little more complicated.

Most arithmetic and logic instructions are of the form

```

OPCODE <ea,Dn
or
OPCODE Dn,<ea

```

where <ea is the effective address, and Dn is a data register. The left most operand is the source operand, and the right is the destination operand. The effective address admits a total of twelve different addressing modes. As we have already seen, not all instructions will permit all of the different addressing modes as either source or destination operands. If the assembler option

OPT PCS

is included at the beginning of a section, all code assembled in that section will automatically be assembled as PCR type operations. Therefore, the code

```
ADD ABLE,D0
```

will be assembled as

```
ADD ABLE(PC),D0
```

However, the code

```
ADD D0,ABLE
```

will be assembled exactly as written because the PCR mode cannot be used for Data Alterable operands. As a result, it is the responsibility of the programmer to replace this code with a sequence like

```
LEA ABLE,A0
ADD D0,(A0)
```

to achieve a position independent equivalent to the above statement. The Load Effective Address, LEA,

admits all PC relative operations. Therefore, this sequence will provide for a Data Alterable operation that is PC relative.

In the operand description

label(PC)

the item label is called a displacement. The effective address is calculated as the sum of the current Program Counter Value and the sign extended sixteen bit value label. All parts of the MC68000 family have a 16 bit limitation to the size of the displacement in this case. For the MC68000 and the MC68010 parts, this limitation exists for the branch instructions as well as the displacements. If it becomes necessary to extend operation beyond the plus or minus 32k byte range dictated by this limitation, other approaches can be used.

Another PC relative addressing operation is the indexed mode. This address mode is described as follows

label(PC,In)

where In is any register. In this case, the address is calculated as the sum of the current Program Counter value, the sign extended value in the register In and the sign extended value of the eight bit label. The displacement in this case is only 8 bits which yields a plus or minus 128 byte range.

It is best to assume that the PCS option will generate satisfactory position independent code when writing assembly language code. The assembler cannot know of all violations of the 32k byte reach of a displacement. This

knowledge becomes evident at link time when link error messages will inform of any such violation. If such an error message occurs during the linking operation, a different approach is needed to create position independent code for the branches or instructions involved.

Position independent jumps can be created. Suppose we wish a jump operation that will move the program control to one of several entries in a table based on a calculated value. Further, suppose that the range to these entries exceed the standard plus or minus 32000 bytes that can be accessed by a branch operation. Any jump operation will require the destination address. Therefore, the code involved must calculate the address of the destination. Examine the following code sequence.

Enter this code sequence with the number of the procedure we wish to execute stored in ENT_NUMB. The table above contains the relative addresses of the several procedures that can be called by this sequence. The first instruction puts the number of the procedure into the data register D3. Each entry in the table is four bytes long. The left shift operation multiplies the pointer value in D3 by 4 to point to the correct location into the table.

Any BSR instruction pushes the value of the return address onto the stack and transfers control of the program to the subroutine address. In this case, we have placed the subroutine as the next instruction following the BSR call. After the BSR instruction is executed, the address of the label HERE will be on top of the stack. The MOVE instruction at

HERE removes this address from the stack and places it in A3. Regardless of the location of this code sequence in memory, the address of HERE will be in A3 at this point. Adding the offset TABLE-HERE to the address will leave the address of TABLE in the address register A3.

The add ADD.L (A3,D3),A3 causes the value stored in the Nth table entry to be added to the address of TABLE. The result of this calculation is the address of PROCN. The JSR instruction then moves control to PROCN.

This fixed code sequence can be moved anywhere in memory and it will execute correctly. Note that the table entries will be calculated at link time if the various procedures are in different modules. Also, it is not even necessary for the table and the

	MOVE.L	ENT_NUMB,D3	Program number is in D3
	ASL.L	D3,2	Multiply pointer by 4
	BSR	HERE	Subroutine call puts
HERE	MOVE.L	(A7)+,A3	address of here in A3
	ADD.L	#TABLE-HERE,A3	Get address of TABLE
	ADD.L	(A3,D3),A3	Calculate the new PROCN
	JSR	(A4)	Go to it
			Subroutine returns here
TABLE	DC.L	PROC0-TABLE	The relative addresses
	DC.L	PROC1-TABLE	of the several proc-
	DC.L	PROC2-TABLE	dures that can be
			called. These values
			are calculated by the
			linker.

FOR THOSE WHO NEED TO KNOW

68 MICRO
JOURNAL™

Bit-Bucket



By: All of us

"Contribute Nothing - Expect Nothing", DMW '86



MOTOROLA INC.

CONTACTS

Zachary Nelson
Cunningham Communication, Inc.
(408) 982-0400

Dean Mosley
Motorola
(512) 440-2839

Bill Bennett
Hewlett-Packard Company
(408) 447-0806

HEWLETT-PACKARD DELIVERS LOW-COST 68030 WORKSTATION

68030 Offers Broad Range of Performance

AUSTIN, TEXAS, Dec. 5. 1988 — Hewlett-Packard Company announced that it will use the Motorola 68030 (030) microprocessor in a new low-cost engineering workstation. The HP 9000 Model 340, that also incorporates Motorola's 68882 (882) math coprocessor, is priced at \$5,495.

Hewlett-Packard has more than 250,000 Motorola-based systems installed worldwide. Earlier this year, Hewlett-Packard added mid-range and high-end workstations based on the 030/882 combination to its 9000 Workstation Family, the HP Model 360 and HP Model 370. The Model 360 provides up to 5 MIPS (million instructions per second) of processing power and the Model 370 provides up to 8 MIPS. The new HP Model 340 complements these products by offering up to 4 MIPS of performance priced at \$10,000 to \$15,000 less than the 360 and 370.

Motorola's 68000 microprocessor line currently has four members — the 68000, 68010, 68020 and 68030. The development of the 68040 continues the evolution of the processor family. In total, 19 million chips from the 68000 family have been sold in applications including supercomputers, high-end workstations, business computers and embedded control devices. New generations of the 68000 are fully compatible with earlier 68000-based products; software written for one chip runs with no modification on the others, and hardware upgrades are very simple.

Hewlett-Packard is an international manufacturer of measurement and computation products and systems recognized for quality and support. The company's products and services are used in industry, business, engineering, science, medicine and education in more than 70 countries. Founded in 1939, the company celebrates its 50th anniversary in 1989. It has 87,000 employees and had revenue of \$8.1 billion in its 1988 fiscal year.

Motorola's \$2.2 billion Semiconductor Products Sector (Phoenix, Ariz.), which includes the Microprocessor Products Group (Austin, Texas), is a part of Motorola Inc. It is the largest and broadest supplier of semiconductors in North America with a balanced portfolio of over 50,000 devices.

AA **MOTOROLA INC.**

EDITORIAL CONTACT

Don Rogers, Motorola
512/441-6022
Or, Ken Phillips
602/952-3637

Semiconductor Products Sector
Public Relations
3102 Nurn 10th Street
Phoenix, Arizona 85018
P.O. Box 53073
Phoenix, Arizona 85072-2873
FROM: Carolyn C. Phillips
(802) 952-3637

A CHIP OFF THE NEW BLOCK

MESA, Ariz., Dec. 2 — In the world of semiconductor manufacturing, Motorola's new MOS 6 high density CMOS wafer fabrication facility at Mesa, Ariz. is the industry's latest crown jewel, one whose products will help usher in a new era.

"This world class facility is Motorola's first U.S. manufacturing site to use some of the Toshiba process technology for manufacturing the latest architectures on six-inch wafers," said Marc Vandenberg, MOS 6 operations manager.

Vandenberg said the 38,000-square-foot, Class 10 factory is dedicated to fabricating one-megabit dynamic random access memories (DRAMs) and other memory circuits, as well as one of the semiconductor industry's hottest products, application-specific integrated circuits (ASICs).

The relationship with Toshiba, a Motorola technology partner and currently the world leader among one-megabit DRAM producers, also positions Motorola to use MOS 6 for future domestic manufacture of four-megabit DRAMs, which are expected to become the dominant memory chip by the early 1990s.

A one-megabit DRAM is capable of storing 1,048,576 bits of information, the equivalent of 64 double-spaced typed pages, on a single integrated circuit. The devices are used extensively in virtually all families of computers and computer-based systems, including computer-aided design, engineering, and manufacturing; communications; robotics; and consumer products.

ASICs, the second product type to be fabricated at the new Mesa facility, are fast becoming the semiconductor equivalent to the customized automobile. Designers select from a set of standard configurations, then add their own features depending on specific needs.

The result is reduced design and production costs and quicker turnaround, with completion of some types of semiconductor circuits in as little as two to three weeks.

Analysts estimate that ASICs, one of the fastest-growing segments of the semiconductor business, accounted for approximately 20 percent of the total market for integrated circuits last year.

Motorola managers said the advanced processes and quick cycle capabilities of MOS 6 are ideal for meeting rapidly changing requirements of high performance customer applications, especially in computers and communications.

The facility is a key element in Motorola's plans for the company's new High Density CMOS (HDC) Series, a line of ASIC products. Capable of processing sub-micron HDC parts as they evolve in the future, MOS 6 provides an immediate capability to manufacture the latest 1.0 micron HDC gate arrays.

Motorola's new ASIC circuits represent a significant advancement in microchip technology. Built with a triple layer metal (TRIM) CMOS process, they give semi-custom chip designers efficient routing and power distribution on a channelless architecture with minimal die sizes.

The result, according to Motorola managers, is a 75 percent gate utilization rate with very high performance (subnanosecond loaded gates) and unprecedented input/output flexibility and density.

"MOS 6 will provide advanced manufacturing technology administered by highly skilled people to support superior customer service and product quality, both in memory products and the rapidly expanding ASIC market," Vandenberg said.



CONTACTS

James Strohecker
Cunningham Communication, Inc.
(408) 982-0400

Bob Anundson
88open Consortium, Ltd.
(503) 682-5703

88open Consortium Holds Member Meeting Completes Binary Compatibility Standard

Two New 88000-based Systems Displayed At The Meetings

WILSONVILLE, Ore. — December 12, 1988 — The 88open Consortium, Ltd. today announced the results of its members meeting held last week, where 128 representatives from more than 50 companies rallied to support the Consortium's standards development and "Team Computing" concept. At the meeting, two hardware vendors displayed yet-to-be-announced computer systems based on Motorola's 88000 RISC microprocessor.

During the two day meeting, held in San Diego, Calif., the 88open Binary Compatibility Standard (BCS) committee met and approved the final BCS that allows 88000 RISC-based application software to operate transparently across numerous systems, similar to PC "shrink wrapped" software.

More than 20 international representatives from European companies met separately at the meeting to review overseas standards development, product announcements and the appointment of an 88open European director. Many of the international representatives indicated that their companies would announce 88000-based hardware and software developments in early to mid 1989.

The meeting provided new and existing 88open members with the opportunity to meet and review progress of software application, hardware and standards development. The members discussed updates on operating system-based groups such as the Open Software Foundation (OSF) and Unix International (formerly the Archer Group), upcoming product announcements and individual and collective publicity efforts.

Chuck Corley, manager of Motorola's 88000 Microprocessor Development, presented a technical and architecture update, including the general sampling schedule and the announcement of improved clock speeds with upcoming versions of the 88000 architecture.

"It was impressive to see that in just six months we evolved from a group of 30 vendors who were reluctant to let each other know which company they represented, to a cohesive representation of more than 125 people who were working together to develop standards and bring about the success of the 88000 architecture," said Bob Anundson, executive director of the 88open Consortium. "We believe that this form of Team Computing is what other RISC vendors must ultimately turn to for true success across the RISC architecture and hardware and software."

The Binary Compatibility Standard is a cornerstone for making the Team Computing environment possible. 88open's concept of Team Computing enables all application software that complies with the BCS to run on any of the 88000-based hardware systems. User benefits include a highly competitive environment with a vast array of cost-effective solutions.

The BCS has been under development by a committee of 88open members since the introduction of the Motorola 88000 RISC architecture in April and is the first RISC standard to be made available in the industry. The 88open BCS is based on AT&T's Unix System V Rel. 3.0 and combines X/Open and IEEE standards. Interested parties may obtain the BCS from 88open.

A number of new members including NCR were added to the 88open roster at the meeting. It was announced that Richard Henter of NCR was appointed a seat on the 88open board of directors. NCR has secretly been a Consortium member since the formal introduction of the 88open in April 1988.

The 88open Consortium, Ltd. is a not-for-profit organization formed to develop and promote the success of Motorola's 88000 RISC microprocessor architecture. The Consortium, based in Wilsonville, Ore., includes more than 51 worldwide members. 88open is pioneering a unique way of doing business built on the portability of 88000-based applications across a variety of hardware platforms including fault tolerant systems, computer servers and workstations. Information regarding the Consortium can be obtained from the 88open Consortium, Ltd. at 8560 SW Salish Lane, Suite 500, Wilsonville, OR 97070; telephone (503) 682-5703.

SANYO/ICON FIRST 88OPEN MEMBER TO DISCLOSE 88000-BASED MULTI-USER SYSTEM

WILSONVILLE, Ore. — December 12, 1988 — The 88open Consortium, Ltd. today announced that Sanyo/Icon International (Orem, Utah), an American subsidiary of the Sanyo Electric Co. (Osaka, Japan), is the first 88open member to introduce a Motorola 88000 RISC processor-based system. The system, called Icon 8000, is a multiuser business system which supports up to 256 users, delivers 15 MIPS (millions instructions per second) of performance and is the first of a number of products from a variety of vendors to comply with the 88000 Binary Compatibility Standard (BCS). The BCS allows 88000 RISC-based application software to transparently operate across numerous systems, similar to PC "shrink wrapped" software.

Sanyo/Icon is one of more than 45 member companies of the 88open. The company joined the 88open Consortium in June of 1988. The 88000 architecture, which includes many features of the so-called "superminicomputer" market, is currently being evaluated by more than 200 manufacturers worldwide.

"The Sanyo/Icon announcement demonstrates the first implementation under a new computing environment we call 'Team Computing,'" said Bob Anundson, executive director of the 88open Consortium. "Team Computing is where all application software will run on any of the 88000-based hardware systems. The benefits to the user are a highly competitive environment with a vast array of cost-effective solutions."

The BCS is a cornerstone for making the Team Computing environment possible. That standard has been under development by a committee of 88open members since the introduction of the 88000 chip set in April, and was the first to be made available in the industry. The 88open BCS is based on AT&T's UNIX System V Rel. 3.0 and combines X/Open and IEEE standards. Interested parties may obtain the BCS from 88open.

The 88open Consortium, Ltd. is a not-for-profit organization formed to develop and promote the success of Motorola's 88000 RISC microprocessor architecture. The Consortium, based in Wilsonville, Ore., includes more than 45 worldwide members. 88open is pioneering a unique way of doing business built on the portability 88000-based applications across a variety of hardware platforms including fault tolerant systems, computer servers and workstations. Information regarding the Consortium can be obtained from the 88open Consortium, Ltd. at 8560 SW Salish Lane, Suite 500, Wilsonville, OR 97070; telephone (503) 682-5703.

EKF-ELEKTRONIK-MESSTECHNIK GMBH
Systemhaus für Mikrocomputer und Industrie-Elektronik



EKF-Elektronik Messtechnik GmbH, Westring 1a, D-4700 Hamm 1

Don Williams
68 Micro Journal
Computer Publishing Center
5900 Cassandra Smith Road

Westring 1a, D-4700 Hamm 1
Telefon (02381) 12630
Telefax (02381) 15067

Data-Path Switchbox for V.24/RS232/RS423 • 68520-SX

Two independent sections of 3-way data communication path selectors for serial interfaces according to EIA RS232 are contained in the 68520-SX switchbox from EKF, thus allowing coupling of several peripheral devices to more than one computer system. Integrated null-modems are provided to simplify changing of DTR to DCE and vice versa, so that connections from computer to computer or terminal to terminal require no more special cabling hardware.

Housed in a steel-sheet box for electrical and magnetical screening, 10 DB-25 female connectors are routed on the pin's Tx-Data, Rx-Data, CTS und DTR (No. 1, 2, 3, 5, 7, 20). Frame (Pin 1) is available via 4mm grounding socket.

Betreiberhinweise:
Friedrich Dortmund (49) 120 46 128 51 487
Deutsche Bank Hamm (49) 120 46 128 51 487
Deutsche Bank Hamm (49) 120 46 128 51 487



Händlerregister Nr. 9 982, Amtsgericht Hamm

EKF-ELEKTRONIK-MESSTECHNIK GMBH
Systemhaus für Mikrocomputer und Industrie-Elektronik



EKF-Elektronik Messtechnik GmbH, Westring 1a, D-4700 Hamm 1

Don Williams
68 Micro Journal
Computer Publishing Center
5900 Cassandra Smith Road

Westring 1a, D-4700 Hamm 1
Telefon (02381) 12630
Telefax (02381) 15067

EKF Adapterboard changes 68020 systems to 68030 CPU • VME 68061-ADP30

With dimensions of only 38x76mm, the EKF adapterboard VME 68061-ADP30 allows upgrading of any existing 68020 microcomputer system to the more advanced 68030 chip. The only thing the user has to do is to remove the 68020 CPU and to insert the adapter, carrying the 68030 microprocessor.

The pc-board is a 8-layer construction with power and groundplane in combination with additional decoupling capacitors for effective noise reduction to insure absolutely reliable operation. Existing 68020 software runs on the 68030 microprocessor without any changes, such that upgraded computers can be booted from harddisk or diskette in the same manner as before. The VME 68061-ADP30 adapter is available from EKF stock at a single quantity price of DM 220.- (Europe) or US\$ 135.- (USA).

Betreiberhinweise:
Friedrich Dortmund (49) 120 46 128 51 487
Deutsche Bank Hamm (49) 120 46 128 51 487
Deutsche Bank Hamm (49) 120 46 128 51 487



50 West Hoover Ave.
Mesa, Arizona 85210
Tel. (602) 962-5559
Fax. (602) 962-5750

GESFAC ISSUES 1989 CATALOG

MESA, AZ, December 9, 1988--GESFAC, Inc has released its new 1989 product catalog of board level products, software and systems. The 116 page, all-color document is the largest the company has ever produced and lists over 300 up-to-date product references.

The GESFAC catalog is the largest in the board level industry and includes the most diversified variety of hardware and software components. The GESFAC catalog offers as much, if not more choice, than the typical bus buyer's guides of other architectures. The GESFAC catalog is sent free of charge to qualified engineers.

The catalog is divided into several major product categories to help guide the engineer's selection. The principal categories are: Intel and Motorola based Processors; static and dynamic Memories; serial, parallel and analog Interfaces; Controllers for disks, graphics, data communication, networking and motor control; various accessories such as card cages, cables and backplanes.

Reader Contact: Don Bizlos
Editorial Contact: Cosma Pebouctsis

A new product category lists a complete offering of linear scan cameras and controllers.

The catalog's software section has grown to include a library of software drivers for most boards offered by GESFAC, software development utilities, in addition to the usual real-time operating systems and high level languages.

An important section of the catalog is dedicated to systems. This section lists a large family of pre-assembled and ready-to-use development systems and OEM computers based on the boards and software developed by the company. These systems are either based on the OMIX-like OS-9 real-time operating system or on the popular MS-DOS operating system.

The catalog also lists Application Specific Integrated Circuits developed by GESFAC for interfacing to the G-64 bus. These ASICs are made available to all third party board manufacturers.



Product News

Prepared By:
Shohet & Kahn PR
2959 S. Winchester Blvd., Campbell, CA 95008
Murry Shohet (408) 379-7434

Contact:
FORCE USA, Wayne Fischer (408) 370-6300
FORCE GmbH, Anton Neusch (089) 600-910

First 68030-based VMEbus Message Passing CPU For Distributed Real-Time Multiprocessing at the High End

CPU-30 Selected For Navy Simulator Program, Displacing
Minicomputers & Mainframes

CAMPBELL, CA., December 13, 1988 — The highest known performance 68030-based VMEbus single board computer began shipping this month. FORCE COMPUTERS believes the CPU-30 sets new marks in functionality and processing throughput that will expand the use of VMEbus products into new markets. The CPU-30 is the first true message passing single board computer available for VMEbus applications. Its introduction begins an era of off-the-shelf solutions for applications previously served by more expensive minicomputers and mainframes.

The CPU-30 has already been selected for use in a major Navy simulator program (see separate release) that stresses these features.

"The CPU-30 enables the design of very large scale high performance systems in which the processing power is distributed among users rather than concentrated in a centralized computer," said Murry Weisberg, FORCE COMPUTERS Vice President & General Manager. "No other VMEbus solution offers the message exchange architecture needed for real-time distributed multiprocessing. The result is much higher performance at lower cost," he added.

Library of Features Makes CPU-30 More Like Minicomputer Than VME CPU

The CPU-30 exceeds the functionality of any known VMEbus board to achieve the status of a true system-on-a-board. Its main features include:

- 68030 microprocessor, 20 or 25MHz operation
- 68882 floating point coprocessor, 20 or 25MHz
- High performance 32-bit DMA controller - high speed data transfer locally and across the VMEbus; 32 byte internal FIFO for burst DMA
- VME/PLUS™ technology implemented in FORCE GATE ARRAY 2 (FGA-002), a 22,000 gate ASIC that provides message passing, mailbox interrupts and a comprehensive VMEbus interface
- 4 Mbytes system memory; shared dynamic RAM with byte parity; accessible from the VMEbus via FGA-002
- Message broadcast to up to 20 CPUs simultaneously
- SCSI interface using on-board DMA controller
- Floppy disk interface (SAA60 compatible)
- Four multiprotocol serial ports: 1 channel is RS232 and 3 are RS232/RS422/RS485 compatible
- 8-bit parallel interface with handshake
- Up to 4 Mbytes of EPROM with support for 28- and 32-pin devices; 32-bit data path
- System EPROM for local boot and smart installation of the I/O interface and FGA-002
- Real-time clock/calendar with battery back-up
- 32 Mbytes of local high-speed SRAM with battery back-up
- Optional 10 Mbit/sec Ethernet controller
- 2 24-bit and 1 8-bit timers
- Full 32-bit VMEbus master/slave interface, single-level arbiter, SYSCLK driver, interrupt handler, support for ACFAIL & SYSFAIL
- VME/PLUS™, FORCE COMPUTERS' standard real-time operating system kernel

Multiprocessing Capability & VME Functions are Second to None

The "crown jewel" of the CPU-30's architecture is FGA-002, a 281 pin ASIC that enables multiprocessing via message passing and mailbox interrupts. FGA-002 also provides all the glue logic needed by the host processor to drive the VMEbus and perform the industry's most comprehensive set of master, slave and control functions. These features are collectively labeled VME/PLUS. On-board performance is boosted because the ASIC minimizes gate delays.

As a multiprocessing engine, the CPU-30 implements message passing modeled on minicomputer and mainframe ideals, but with VMEbus compatibility. Message passing permits any processor to broadcast, at any time, status, data, interrupt or other messages to a select but potentially large set of receivers. Because message passing enables real time system operation, a backplane populated with several VME/PLUS CPUs can handle a vast array of functions without contention or system crashes.

The CPU-30 employs an 8-bit wide, 8-byte deep FIFO message buffer and an 8-bit wide high priority message register. The byte-wide configuration enables a scheme of 256 possible messages. Any VME master can deposit a message in the buffer or register. Up to 20 VME/PLUS-equipped boards can simultaneously receive a message. Message receipt stimulates quick action, typically on a predetermined basis. For example, servicing the interrupt caused by message receipt may turn-off an alarm, move a robot arm, calculate a block of data or send another message to another CPU, etc. The action taken is totally software programmable. It is easily tailored to specific applications. And, it happens more swiftly and with greater certainty of success than any other architectural approach that complies with the IEEE-1014 industry-standard VMEbus specification.

Directed Software Interrupts via Mailboxes

The VMEbus provides 7 hardware interrupt levels; this restricts the number of active interrupt handlers to 7 at a time. The CPU-30 augments this by providing 8 Mailbox Interrupts which are software driven. With 8 per board, a fully populated backplane (21 CPUs) provides 168 directed software interrupts for handling the interrupt volume expected in complex systems. The use of Mailboxes allows swift execution through quick generation of interrupts, and results in incredible flexibility.

Corporate News

Prepared By:
Shohet & Kahn PR
2959 S. Winchester Blvd., Campbell, CA 95008
Murry Shohet (408) 379-7434

Contact:
FORCE USA, Wayne Fischer (408) 370-6300
CAE-LINK, Bob Teggart (301) 622-4400

FORCE VME/PLUS™ Technology Chosen by CAE-LTSD for Navy Simulator

Largest Contract in FORCE History Also Marks Penetration of
VMEbus into Minicomputer and Mainframe Territory

ORLANDO, FL., November 30, 1988 — Link Tactical Simulation Division (LTSD), located in Silver Spring, MD, (a subsidiary of CAE Industries located in Toronto, Ontario, Canada) and FORCE COMPUTERS (Campbell, CA) have announced award of a major subcontract for computer subsystems for the U.S. Navy's 14A12 Surface ASW (Anti-Submarine Warfare) Trainer.

CAE-Link is the prime contractor for the production 14A12. This subcontract with FORCE will result in the supply of very high performance VMEbus-based computers that provide both student and instructor stations in an elaborate, extremely realistic re-creation of surface ASW strategies and tactics.

Under terms of the contract, FORCE is implementing the CAE-Link design and delivering fully integrated, functional subsystems.

The announcement was made on the eve of the annual UTSC Conference (Interuniversity Training Service Conference), held here. At its exhibit booth, CAE-Link will demonstrate working student stations developed with FORCE during a development phase. FORCE also plans to demonstrate working models of the VMEbus computer design used in the program. It will be the first public showing of these extremely powerful computers.

The 14A12 ASW Trainer is an aggressive Navy program to build an ASW training capability for the crews of surface ships. The program has an expected procurement life of several years, during which the FORCE subcontract could be worth more than \$10 million. Delivery of the FORCE hardware begins in early 1989.

FORCE Chosen For Product Performance, Design Synergy

Bob Teggart, CAE-Link's Vice President of Marketing, said "the FORCE VMEbus-based solution is a sharp departure from the minicomputers that usually get the nod for this type of work. VMEbus hardware will change the face of simulation because it's inherently more powerful, easy to distribute where needed, consumes much less power and has much lower life cycle cost. This approach results in modular building blocks that can be used in other trainers. We chose FORCE because their latest VME technology offers much higher performance and the ability to distribute processing power and multiprocessing capability in a complex simulation involving many students and instructors. Their real-time hardware is not only more powerful, it's also less expensive."

Teggart also indicated that FORCE was chosen over several competitors "because they joined our design team and provided the ongoing support and synergy that helped us to win the Navy contract."

Many Weisberg, FORCE Executive Vice President and General Manager, said "CAE-Link's leadership in simulation continues because they were willing to try something new. FORCE is using its latest design for the 14A12. These boards employ the 68030 microprocessor in an exciting new architecture called VME/PLUS that provides mainframe computer qualities such as message passing and mailbox interrupts. We think this contract marks an era in simulation and training procurement using off-the-shelf industry-standard solutions."

During the year-long development phase of the 14A12, FORCE provided hardware and also prepared a real-time operating system, rSOS.

Trainers Will Enhance Readiness

The 14A12 Surface ASW Trainer has the potential to improve combat readiness. The Navy's plan call for construction of several 14A12 Trainers. Weisberg said the use of VME/PLUS distributed multiprocessing in the 14A12 design was being looked at closely as a candidate for other simulation and training programs across the military services and commercial airlines.

About Force Computers

The leading independent designer and manufacturer of VMEbus products, Force is entering its seventh year. The company has completed 23 consecutive quarters of profitable operation. Force is headquartered in Campbell, California with subsidiaries in West Germany, France and the United Kingdom. Sales, service and product support are provided on a worldwide basis.

About CAE-Link

Link Tactical Simulation Division is the leading producer of training simulators for the U.S. Navy's surface, air and subsurface ASW applications; surface team tactics training and ground C² systems for the U.S. Army.

Dear Don,

A preview of friend Denis' 6013 for December shows a few typos, which readers may wish to correct. These are:

P16, last para should read "What does S₁ ABC' convey"
Diagram 85, row 3, col 10 should read 120, not 126.
P20, last para but one, should read "If they're equal,
we get directed to BLOC:-4a ..."
P22, para 6, should read "... function as S₁ ABC' ..."

I've also noticed a sneaky one in the August issue, page 21, near top, should read n2 m1 n2 n1. The n2/n1 combo somehow got reversed!

I've been asked by a few readers about Manuals for RBASIC. The price for the manual alone is US\$25, including postage for North America (plus \$5 elsewhere), \$22 of which will be applied against a future purchase of RBASIC.

Don Williams,
68 Micro Journal,
5900 Cassandra Smith Road,
Hixson, TN 37343

Sincerely,



R. Jones
President

Classifieds As Submitted - No Guarantees

MUSTANG-020 16Mhz with 68881. OS9 Professional Package & C \$2500.

S+System with Cabinet, 20 Meg Hard Disk & 8" Disk Drive with DMAF3 Controller Board. 1-X12 Terminal \$2900.

HARD DISK 10 Megabyte Drive - Seagate Model #412 \$275.

3-Dual 8" drive enclosure with power supply. New in box. \$125 each.

5-Siemens 8" Disk Drive, \$100 each.

SWTPC S/09 with Motorola 128K RAM, 1-MPS2, 1-Parallel Port, MP-09CPU Card- \$490 complete.

Tom (615) 842-4600 M-F 9AM to 5PM EST

Motorola VME-10 with hi resolution monochrome monitor, hard disk, serial and parallel cards, Pascal, assembler, linker and documentation. Almost new \$4500.

Two SSB-6809 systems with hard disks, miscellaneous software. Make Offer

Cadwell Laboratories
909 N. Kellogg Street
Kennewick, WA 99336
(509) 735-6431

BICC VERO MICROSYSTEMS
Flanders Road, Hedge End,
Southampton, Hampshire.
SO3 3LG. ENGLAND

Tel: 0703 266300

BICC VERO MICROSYSTEMS provide a range of OS9/VME boards and Development Systems all with full OS9 drivers. A number of development tools such as Verolink are also available to simplify development of ROM based target systems. In addition 'C' bindings are available for Graphics, GPIB, Analogue I/O and Parallel I/O.

B&R Industrial Automation Corp. is a manufacturer of programmable logic controllers, industrial computers, visualization and process control systems. B&R offers the own hardware and software with easy communication to other systems.

One of the latest developments is a OS9 based minicomputer system with floppy controller on board, external harddisk (40 MB), optional memory extension board (RAM and EPROM) and a high resolution graphic controller. This so-called MAESTRO-System is available for OS9 based software or can be used with B&R standard software and offers direct access to digital and analog inputs and outputs.

For more information please contact: B&R Industrial Automation Corp.
2165 West Park Court
Stone Mountain, GEORGIA 30087
Tel: 404 469 4617, Fax: 404 469 5460

H.C. ANDERSEN COMPUTER A/S Th. Philipsensvej 21-23 DK-2770 Kastrup, Denmark Telephone: 45 1 52 44 04 Telex: 31484 tmrcph	License: OS9/6809 Level 1 (DRAGON Computer) Development: OS9/68000 ATARI ST (Dr.Keil) OS9/68000 ATARI ST (Cumana) Drivers for t2->t5, cadscrn /term with special fonts, TOS GATEWAY (PD) SCULPTOR development
--	--

OS-9 for VME: Single Boards to Complete Systems

Mizar provides complete OS-9 solutions for the VMEbus. Mizar's VME CPUs offer the functions and performance your application demands. Our single-height (3U) VME processors are uniquely configurable computing engines. Through Mizar's unique MXbus™ expansion interface, standard and custom side modules can be added to basic processors to create double-height (6U) boards for specific applications. 3U CPU options include 68010, 68020, and 68030 microprocessors, up to one MB of DRAM, serial I/O, real-time clock, and mailbox interrupt support. Standard MXbus side modules include additional DRAM, SRAM, and I/O.

Mizar's standard double-height (6U) processors provide additional features such as a high-speed cache to enhance 68030 performance, floating point coprocessor support, up to four MB dual-ported DRAM, VSB memory interface, Ethernet, and SCSI.

Mizar also supports OS-9 with completely configured OS-9 development systems and OS-9 application server systems.

For more information, call Mizar today.

800-635-0200
MIZAR

1419 Dunn Drive • Carrollton, TX 75006 • 214-446-2664

Pan Controls Limited specifically provides services in the field of industrial automation. OS9 is used for the majority of the high level computing work, and a number of software packages have been developed for sale generally.

STIMULUS is a general purpose rule-based expert system programming language designed for easier and more economic real-time processing for engineering control, simulation, monitoring, diagnosis, etc. Since STIMULUS produces C source, it may be linked with other modules in C or Assembly language, giving you the choice of C's I/O and interrupt facilities.

PROFILE is a symbol profiling tool for C or Assembly language programs that increases programmer efficiency. It provides a valuable and quick method of optimising large and small projects by making a table of the time spent running each of your functions in your programme during a working session. A quick look at this profile table shows you which functions are most used and help you to concentrate your programming efforts on optimising these functions.

PAN UTILITIES is a library of programs providing the major text-handling, file-handling and programmers utilities which the OS9 operating system lacks.

PEP Modular Computers is one of the leading manufacturers of microcomputer boards and systems for the VMEbus and IIOC (Intelligent I/O Channel) in compact single height format./Founded in 1975 in Kaufbeuren, Germany, PEP is now established worldwide with subsidiaries in USA, France and Sweden./Today PEP offers a broad range of VME and IIOC board level products, plus development systems, racks, backplanes, power supplies and standard operating systems./ PEP's target market segments are industrial control applications including machine control, process control, data acquisition and robotics. This year PEP introduced for example VSCSI, a SCSI interface module for the VMEbus, VDAO, a combined A/D, D/A and digital I/O VME single height module, VMPM68KC-2, a powerful 68020 VMEbus CPU in military temperature range, VLAN, a cost-effective VME networking solution, VGPM a high performance graphics module based on the Hitachi ACRTC63484. The company also introduced VIOXROM, an universal VME software interface for a family of intelligent VME I/O modules./ Currently about 100 people are working for PEP worldwide. For further information please contact: Germany 08341/81001

Scorpion Technologies Inc. features a line of Motorola 680X0 based co-processor add-in boards for IBM PC/XT/AT and compatibles providing the most economical development platform for OS-9 engineers and programmers. This is the only known implementation of Professional OS-9 available on the PC platform and provides the unique feature of PC DOS concurrency. The Pro68 family is available with choice of 68000, 68010 or 68020 processors and a minimum of 1MB of memory (expandable to 4MB). Two serial ports are available on the 68010 and 68020 versions with support for 16 users via intelligent serial I/O cards.

UNIVERSAL ETHERNET CONTROLLER BOARD MPVME1054

* Intelligent Ethernet Slave Board running in any VME System

A TRUE MICROCOMPUTER BOARD SOLUTION WITH ETHERNET CAPABILITY

MPVME 1002 (16 Bit) MPVME1021 (32 Bit)

* Three communication packages under OS-9/68K
for all three Board available:

- DECNET/VME
- TCP/IP
- OS9-NET

Call for a complete documentation

SYSTEMFORSCHUNG, Königstr. 33a, 5300 Bonn 1, W-Germany Tel. 0228/223151

XYZ Electronics, Inc. RR 12, Box 322 Indianapolis, IN 46236
(317) 335-2128 Contact: Gary Bannister

10-year old company manufacturers STD bus based industrial computer systems and boards/ Suport OS-9/6809 Level I and OS-9/68000 Professional Package.

Board level products include CPU-9A 2MHz 68B09, and CPU-68K3 10 MHz 68009. Both boards support STD bus for memory and I/O expansion. Board level products, board sets, and complete systems available. Memory, Floppy and hard disk controllers, analog I/O, digital I/O, and cages available.

STD Bus systems offer a superior price/performance ration where medium level performance in an industrial environment is required.

NEW!

OmegaSoft Pascal for the 68020/68881

P20K is a Pascal package that will generate code for all of the 68000 series processors, including the 68881 coprocessor. P20K will run on any 68000 series computer running the OS-9/68000 (Microware) or PDOS (Eyring Research) operating systems with 512K or more free memory.

The base package (P20K-B) includes the Compiler, Relocatable Macro Assembler, Linking Loader, Screen Editor, Pascal Shell, Linkage Creator, Host Debugger, Configuration manager, Installation program, and Patch utility. A new feature in this compiler is the ability to either link in the parts of the runtime needed by the program, or to use trap handlers for runtime access, to share the runtime library between programs. Complete operating system interface is also included using Pascal procedures and functions. The host debugger allows debugging at both the Pascal and assembly language levels of programs that run on the host operating system. Price for the base package is \$575.

The runtime source code option (P20K-R) is available for \$100 and includes source code for the operating system interface routines as well as Pascal runtime.

The Utility source option (P20K-S) is available for \$275 and includes source code for the Screen Editor, Pascal Shell, Host Debugger, Patch utility, and Configuration manager.

The Target debugger option (P20K-T) is \$225 and includes object and source code. This program allows Pascal level and assembly level debugging in a system without operating system, by using a serial link connected to the host computer.

Prices do not include shipping charges. Master-Card and Visa accepted. OmegaSoft is a registered trademark of Certified Software Corporation.

Gespac SA, 3, Chemin des Aulx, CH-1228,
Geneva/Plan-les-Quatre, Switzerland
TEL 022-713400, TLX 429989

Elsoft AG, Zeilweg 12, CH-5405 Baden-Dättwil,
Switzerland. TEL 056-833377, TLX 828275

RCS Microsystems Ltd, 141 Uxbridge Road,
Hampton Hill, Middlesex, England
TEL 01-9792204, TLX 8851470

Byte Studio Borken, Butenwall 14, D-4280 Borken,
West Germany.
TEL 02861-2147, TLX 813343

Eltec Elektronik GmbH, Galileo-Galilei-Straße,
6500 Mainz 42, Postfach 65, West Germany
TEL 06131-50031, TLX 4187273

PEP Elektronik Systeme GmbH, Am Klosterwald 4,
D-8950 Kaufbeuren, West Germany
TEL 08341-8974, TLX 541233

**CERTIFIED
SOFTWARE
CORPORATION**

P.O. BOX 70, RANDOLPH, VT 05060 USA
TELEPHONE: (802) 728-4062
FAX: (802) 728-4126

FLEX™/SK-DOS™/MS-DOS™

Transfer Utilities

For 68XXX and CoCo* OS-9™ Systems
Now READ - WRITE - DIR - DUMP - EXPLORE
FLEX, SK-DOS & MS-DOS Disk

These Utilities come with a rich set of options allowing the transfer of text type files from/to FLEX & MS-DOS disks.

*CoCo systems require the D.P. Johnson SDISK utilities and OS-9 and two drives of which one must be a "host" floppy.

CoCo Version: \$69.95

68XXX Version \$99.95

S.E. Media

615 842-6809

PO Box 849, Hixson, TN 37343

MC/Visa

SK*DOS®/68K

Read the fine print to see what's in SK*DOS/68K:

☐ Full DOS documentation plus on-line help ☐ Multiple directories
☐ User-installable device drivers ☐ Install up to 8 different I/O devices ☐ Keyboard type-ahead ☐ Print-screen ☐ Virtual (RAM) disk ☐ Disk cache ☐ Up to 10 drives ☐ 5¼" or 3½" floppy drives ☐ Hard drives to 64 megabytes each ☐ I/O redirection to drives or I/O ☐ Time/date stamping of files ☐ File or disk write protect (even hard disk) ☐ Batch files ☐ Support for 68000, 68010, 68020 ☐ Monochrome or color video board support ☐ Read and write MS-DOS disk files ☐ 6809 Emulator ☐ Powerful utilities such as copy-by-date, undelete, show differences between files, prompted delete, text file browse, and more - all included ☐ Simple Basic included ☐ Fast assembler included ☐ Line editor included ☐ User support via newsletter and BBS ☐ Available software: C compiler, full Basic, screen editors, disassemblers, cross-assemblers, spelling checker, text formatter, music editor, hard disk manager, ROM-based debugger, modem communications programs, etc. More compilers coming. ☐ (Some features may not be implemented in all hardware manufacturers' implementations.)

Individual copies of SK*DOS/68K are \$140; less in quantity or when bundled with hardware. Send for our 6809 / 68K hardware and software catalog. Also available as part of our hardware/software educational course.



Software Systems Corp.
 P. O. Box 209J
 Mt. Kisco, NY 10549
 (914) 241-0287
 BBS (914) 241-3307 ♦ Fax (914) 241-8607



APPLE

MACINTOSH™



USERS

Save over a \$1,000.00
on PostScript
Laser Printers!
Faster - Finer Quality
than the original Apple
LaserWriter!
New & Demos
Cartridges-new-rebuilds
-colors-

In Chattanooga Call:
615 812-4600
QMS-Authorized

Data-Comp Division
 A Decade of Quality Service
 Systems World Wide
 Computer Publishing, Inc. 5800 Casserra Smith Road
 Telephone 615-842-4801 • Telex 510 600-6630 Hickory, TN 37343

SOFTWARE

68000 C CROSS-COMPILER

\$100 - SKDOS.MSDOS.UNIX.XENIX (OBJECT ONLY)

Accepts K&R C language, generates 68000 assembler code; includes 68010 cross-assembler; libraries provided for SKDOS, but may be modified.

CROSS-ASSEMBLERS WITH MACRO CAPABILITIES

EACH \$50-FLEX.OS9.UNIFLEX.MSDOS.UNIX.SKDOS.XENIX 315100 A1.1/\$100

Specify: 180x, 6502, 68010/11, 6804, 6805, 6809, Z8, Z80, 8048, 8051, 8085, 68010, 32000
 Modules cross-assemblers in C, with load/unload utilities. Sources for additional \$50 each, \$100 for 3, \$300 for all

C/MODEM TELECOMMUNICATIONS PROGRAM

\$100-MSDOS.SKDOS.UNIX.FLEX.XENIX.UNIFLEX OBJECT-ONLY: EACH \$50

Micro-driven with terminal mode, file transfer, MODEM7, XON/XOFF, etc.

SUPER SLEUTH DISASSEMBLERS

EACH \$99-FLEX \$101-OS9 \$100-UNIFLEX OBJECT-ONLY: EACH \$50 FLEX.OS9.COCO

Interactively generate source on disk with labels, includes xref, binary editing
 Specify 6800, 1.2.3.5.8.9/6502 version or Z80/8080.5 version
 COCO DOS available in 6800, 1.2.3.5.8.9/6502 version (not Z80/8080.5) only
 48010 version \$100-FLEX.OS9.UNIFLEX.MSDOS.UNIX.SKDOS.XENIX

DEBUGGING SIMULATORS FOR POPULAR 8-BIT CPUs

EACH \$75-FLEX \$100-OS9 \$80-UNIFLEX OBJECT-ONLY: EACH \$50-COCO FLEX, COCO OS9

Interactively simulate processors, includes disassembly, formatting, binary editing
 Specify for 6800/1, 14x805, 6502, 6809 OS9 only, Z80 FLEX only

ASSEMBLER CODE TRANSLATORS FOR 6502, 6800/1, 6809

6502 to 6809 \$75-FLEX \$85-OS9 \$80-UNIFLEX
 6800/1 to 6809 & 6809 to pos.ind. \$50-FLEX \$75-OS9 Only \$60-UNIFLEX

FULL-SCREEN X BASIC PROGRAMS with cursor control AVAILABLE FOR FLEX, UNIFLEX, AND MSDOS

Display Generators / Documentation	\$30 w/source, \$25 without
Mailing List System	\$100 w/source, \$50 without
Inventory with MRP	\$100 w/source, \$50 without
Table Resa Spreadsheet	\$100 w/source, \$50 without

DISK AND X BASIC UTILITY PROGRAM LIBRARY \$30-FLEX \$30-UNIFLEX/MSDOS

Edit disk sectors, sort directory, maintain master catalog, do disk sorts, resequence some or all BASIC programs, use BASIC program, etc. non-FLEX versions include sort and resequence only

PROFESSIONAL SERVICES FOR THE COMPUTING COMMUNITY

CUSTOMIZED PROGRAMMING

We will customize any of the programs described in this advertisement or in our brochure for specialized customer use or to cover new processors; the charge for such customization depends upon the complexity of the modifications.

CONTRACT PROGRAMMING

We will create new programs or modify existing programs on a contract basis, a service we have provided for over twenty years; the computers on which we have performed contract programming include most popular models of mainframes, including IBM, Burroughs, Univac, Honeywell, most popular models of minicomputers, including DEC, IBM, DG, HP, AT&T, and most popular brands of microcomputers, including 6800/1, 6809, Z80, 6502, 680x0, using most appropriate languages and operating systems, on systems ranging in size from large telecommunications to single board controllers; the charge for contract programming is usually by the hour or by the task.

CONSULTING

We offer a wide range of business and technical consulting services, including seminars, advice, training, and design, on any topic related to computers; the charge for consulting is normally based upon time, travel, and expenses.

Computer Systems Consultants Inc.

1454 Latta Lane
 Caryville, Georgia 30207
 (404) 483-4570 • (404) 483-1717

Contact us about catalog, dealer, discounts, and services. Most programs in source: give computer, OS, disk size, 25% off multiple purchases of same program on one order. VISA and MASTER CARD accepted. Add GA sales tax (if in GA) and 5% shipping. (UNIFLEX to Technical Systems Consultants; OS9 Microware COCO Tandy; MSDOS Microsoft; SKDOS Stack Software)

Clearbrook Software Group

(604)853-9118



CSG IMS is *THE* full featured relational database manager for OS9/OSK. The comprehensive structured application language and B + Tree index structures make CSG IMS the ideal tool for file-intensive applications.

CSG IMS for CoCo2/3 OS9 L1/2 (single user)	\$169.95
CSG IMS for OS9 L2 or 68000 (multi user)	\$495.00
CSG IMS demo with manual	\$30

MSF - MSDos File Manager for CoCo 3/OS9 Level 2

allows you to use MSDos disks directly under OS9.
Requires CoCo 3, OS9 L2, SDISK3 driver \$45.00

SERINA - System Mode Debugger for OS9 L2

allows you to trace execution of any system module, set break points, assemble and disassemble code and examine and change memory.

Requires CoCo3 or Gimix II, OS9 L2 & 80 col. terminal \$139.00

ERINA - Symbolic User Mode Debugger for OS9

lets you find bugs by displaying the machine state and instructions being executed. Set break points, change memory, assemble and disassemble code.

Requires 80 column display, OS9 L1/2 \$69.00

Shipping: N. America - \$5, Overseas - \$10
Clearbrook Software Group P.O. Box 8000-499, Sumas, WA 98295
OS9 is a trademark of Microware Systems Corp., MSDos is a trademark of Microsoft Corp.

SPECIAL

ATARI™

&

OS-9™

NOW!

If you have either the
Atari 520 or 1040 -
you can take
advantage of the
"bargain of a lifetime"
OS-9 68K and BASIC
all for the low, low price of:

\$150.00

Call or Write

S.E. Media

5900 Cassandra Smith Rd.

Hixson, TN 37343

615 842-4601

ATARI & AMIGA CALL

As most of you know, we are very sensitive to your wishes, as concerns the contents of these pages. One of the things that many of you have repeatedly written or called about is coverage for the Atari & Amiga™ series of 68000 computers.

Actually we haven't been too keen on those systems due to a lack of serious software. They were mainly expensive "game-toy" systems. However, recently we are seeing more and more honest-to-goodness serious software for the Atari & Amiga machines. That makes a difference. I feel that we are ready to start some serious looking into a section for the Atari & Amiga computers. Especially so since OS-9 is now running on the Atari (review copy on the way for evaluation and report to you) and rumored for the Amiga. Many of you are doing all kinds of interesting things on these systems. By sharing we all benefit.

This I must stress - Input from you on the Atari & Amiga. As most of you are aware, we are a "contributor supported" magazine. That means that YOU have to do your part. Which is the way it has been for over 10 years. We need articles, technical, reviews of hardware and software, programming (all languages) and the many other facets of support that we have pursued for these many years. Also I will need several to volunteer to do regular columns on the Atari & Amiga systems. Without constant input we can't make it fly! So, if you do your part, we certainly will do ours. How about it, drop me a line or give me a phone call and I will get additional information right back to you. We need your input and support if this is to succeed!

DMW

THE 6800-6809 BOOKS

..HEAR YE.....HEAR

OS-9™ User Notes

By: Peter Dibble

The publishers of 68' Micro Journal are proud to make available the publication of Peter Dibble's

OS9 USER NOTES

Information for the BEGINNER to the PRO,
Regular or CoCo OS9

Using OS9

HELP, HINTS, PROBLEMS, REVIEWS, SUGGESTIONS, COMPLAINTS,
OS9 STANDARDS, Generating a New Bootstrap, Building a
new System Disk, OS9 Users Group, etc.

Program interfacing to OS9

DEVICE DESCRIPTORS, DIRECTORIES, "FORKS", PROTECTION,
"SUSPEND STATE", "PIPES", "INPUT/OUTPUT SYSTEM", etc.

Programming Languages

Assembly Language Programs and Interfacing; Basic09, C,
Pascal, and Cobol reviews, programs, and uses; etc.

Disks Include

No typing all the Source Listings in. Source Code and,
where applicable, assembled or compiled Operating
Programs. The Source and the Discussions in the
Columns can be used "as is", or as a "Starting Point"
for developing your OWN more powerful Programs.
Programs sometimes use multiple Languages such as a
short Assembly Language Routine for reading a
Directory, which is then "piped" to a Basic09 Routine
for output formatting, etc.

BOOK \$9.95

Typeset -- w/ Source Listings
(3-Hole Punched; 8 x 11)

Deluxe Binder - - - - - \$5.50

All Source Listings on Disk

1-8" SS, SD Disk - - - - \$14.95

2-5" SS, DD Disk - - - - \$24.95

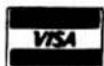
Shipping & Handling \$3.50 per Book, \$2.50 per Disk set

Foreign Orders Add \$4.50 Surface Mail
or \$7.00 Air Mail

If paying by check - Please allow 4-6 weeks delivery

* All Currency in U.S. Dollars

Continually Updated In 68 Micro Journal Monthly



Computer Publishing Inc.
5900 Cassandra Smith Rd.
Hixson, TN 37343



*FLEX is a trademark of Technical Systems Consultants
*OS9 is a trademark of Microware and Motorola
*68' Micro Journal is a trademark of Computer Publishing Inc.

FLEX™ USER NOTES

By: Ronald Anderson

The publishers of 68 MICRO JOURNAL are proud to make available the publication of Ron Anderson's **FLEX USER NOTES**, in book form. This popular monthly column has been a regular feature in 68' MICRO JOURNAL SINCE 1979. It has earned the respect of thousands of 68 MICRO JOURNAL readers over the years. In fact, Ron's column has been described as the 'Bible' for 68XX users, by some of the world's leading microprocessor professionals. The most needed and popular 68XX book available. Over the years Ron's column has been one of the most popular in 68 MICRO JOURNAL. And of course 68 MICRO JOURNAL is the most popular 68XX magazine published.

Listed below are a few of the **TEXT** files included in the book and on diskette.

All TEXT files in the book are on the disks.

LOGO C1	File load program to offset memory — ASM PIC
MEMOVE C1	Memory move program — ASM PIC
DUMP C1	Printer dump program — uses LOGO — ASM PIC
SUBTEST C1	Simulation of 6800 code to 6809, show differences — ASM
TERM C2	Modem input to disk (or other port input to disk) — ASM
M C2	Output a file to modem (or another port) — ASM
PRINT C3	Parallel (enhanced) printer driver — ASM
MODEMC2	TTL output to CRT and modem (or other port) — ASM
SCIPKG C1	Scientific math routines — PASCAL
U C4	Mini-monitor, disk resident, many useful functions — ASM
PRINT C4	Parallel printer driver, without PFLAG — ASM
SET C5	Set printer modes — ASM
SETBAS1 C5	Set printer modes — A-BASIC

NOTE: .C1, .C2, etc. = Chapter 1, Chapter 2, etc.

**Over 30 TEXT files included in ASM (assembler)-PASCAL-PIC (position independent code) TSC BASIC-C, etc.

Book only: **\$7.95** + \$2.50 S/H

With disk: 5" **\$20.90** + \$2.50 S/H

With disk: 8" **\$22.90** + \$2.50 S/H

(615) 842-4601
Telex 5108008630

!!! Subscribe Now !!! 68 MICRO JOURNAL

OK, PLEASE ENTER MY SUBSCRIPTION

Bill My: Mastercard ☐ VISA ☐

Card # _____ Exp. Date _____

For 1 Year ____ 2 Years ____ 3 Years ____

Enclosed: \$ _____

Name _____

Street _____

City _____ State _____ Zip _____

Country _____

My Computer Is: _____

Subscription Rates

U.S.A.: 1 Year \$24.50, 2 Years \$42.50, 3 Years \$64.50

*Foreign Surface: Add \$12.00 per Year to USA Price.

*Foreign Airmail: Add \$48.00 per Year to USA Price.

*Canada & Mexico: Add \$9.50 per Year to USA Price.

*U.S. Currency Cash or Check Drawn on a USA Bank !

68 Micro Journal
5900 Cassa dra Smith Rd.



POB 849
Hixson, TN 37343



Telephone 615 842-4600
Telex 510 600-6630

Reader Service Disks

- Disk- 1 Fileson, Minicat, Minicopy, Minifms. **Lifetime. **Poetry, **Foodlist, **Diet.
- Disk- 2 Diskedit w/ inst.& fixes, Prime, *Pmod, **Snoopy, **Football, **Hexpaw, **Lifetime.
- Disk- 3 Cbug09, Sec1, Sec2, Find, Table2, Intext, Disk-exp, *Disksave.
- Disk- 4 Mailing Program, *Finddat, *Change, *Testdisk.
- Disk- 5 *DISKFIX 1, *DISKFIX 2, **LETTER, **LOVESIGN, **BLACKJAK, **BOWLING.
- Disk- 6 **Purchase Order, Index (Disk file indx).
- Disk- 7 Linking Loader, Rload, Harkness.
- Disk- 8 Crttest, Lanpher (May 82).
- Disk- 9 Datecopy, Diskfix9 (Aug 82).
- Disk-10 Home Accounting (July 82).
- Disk-11 Dissembler (June 84).
- Disk-12 Modem68 (May 84).
- Disk-13 *Inimf68, *Testmf68, *Cleanup, *Diskalign, Help, Date, Txt.
- Disk-14 *Init, *Test, *Terminal, *Find, *Diskedit, Init Lib.
- Disk-15 Modem9 + Updates (Dec. 84 Gilchrist) to Modem9 (April 84 Commo).
- Disk-16 Copy.Txt, Copy.Doc, Cat.Txt, Cat.Doc.
- Disk-17 Match Utility, RATBAS, A Basic Preprocessor.
- Disk-18 Parse.Mod, Size.Cmd (Sept. 85 Armstrong), CMDCODE, CMD.Txt (Sept. 85 Spray).
- Disk-19 Clock, Date, Copy, Cat, PDEL.Asm & Doc., Errors.Sys, Do, Log.Asm & Doc.
- Disk-20 UNIX Like Tools (July & Sept. 85 Taylor & Gilchrist), Dragon.C, Grep.C, LS.C, FDUMP.C.
- Disk-21 Utilities & Games - Date, Life, Madness, Touch, Goblin, Starshot, & 15 more.
- Disk-22 Read CPM & Non-FLEX Disks. Fraser May 1984.
- Disk-23 ISAM, Indexed Sequential file Accessing Methods, Condon Nov. 1985. Extensible Table Driven. Language Recognition Utility, Anderson March 1986.
- Disk-24 68' Micro Journal Index of Articles & Bit Bucket Items from 1979 - 1985, John Current.
- Disk-25 KERMIT for FLEX derived from the UNIX ver. Burg Feb. 1986. (2)-5" Disks or (1)-8" Disk.
- Disk-26 Compacta UniBoard review, code & diagram, Burlison March '86.
- Disk-27 ROTABIT.TXT, SUMSTEST.TXT, CONDATA.1XT, BADMEN.TXT.
- Disk-28 CT-82 Emulator, bit mapped.
- Disk-29 **Star Trek
- Disk-30 Simple Winchester, Dec. '86 Green.
- Disk-31 *** Read/Write MS/PC-DOS (SK-DOS)
- Disk-32 Heir-UNIX Type upgrade - 68MJ 2/87
- Disk-33 Build the GT-4 Terminal - 68MJ 11/87 Condon.
- Disk-34 FLEX 6809 Diagnostics, Disk Drive Test, ROM Test, RAM Test - 68MJ 4/88 Koipi.

NOTE:

This is a reader service ONLY! No Warranty is offered or implied, they are as received by 68' Micro Journal, and are for reader convenience ONLY (some MAY include fixes or patches). Also 6800 and 6809 programs are mixed, as each is fairly simple (mostly) to convert to the other. Software is available to cross-assemble all.

* Denotes 6800 - ** Denotes BASIC

*** Denotes 68000 - 6809 no indicator.



8" disk \$19.50
5" disk \$16.95



Shipping & Handling -U.S.A. Add: - \$3.50
Overseas add: \$4.50 Surface - \$7.00 Airmail

68 MICRO JOURNAL

5900 Cassandra Smith Rd.
Hixson, TN 37343
(615) 842-4600 - Telex 510 600-6630

K-BASIC™

The Only 6809 BASIC to Binary Compiler for OS-9
FLEX or SK*DOS

Even runs on the 68XXX SK*DOS Systems*

*Hundreds Sold at
Suggested Retail:*

~~\$199.00~~

• 6809 - OS-9™ users can now transfer their FLEX™ Extended BASIC (XBASIC) source files to OS-9, compile with the OS-9 version and run them as any other OS-9 binary "CMD" program. Much faster than BASIC programs.

• 6809 - FLEX users can compile their BASIC source files to a regular FLEX ".CMD" file. Much faster execution.

• 68XXX - SK*DOS™ users running on 68XXX systems (such as the Mustang-08/A) can continue to execute their 6809 FLEX BASIC and compiled programs while getting things ported over to the 68XXX. SK*DOS allows 6809 programs to run in emulation mode. This is the only system we know of that will run both 6809 & 68XXX binary files.

K-BASIC is a true compiler. Compiling BASIC 6809 programs to binary command type programs. The savings in RAM needed and the increased speed of binary execution makes this a must for the serious user. And the price is now RIGHT!

Don't get caught up in the "Learn a New Language" syndrome - Write Your Program In BASIC, Debug it In BASIC and Then Compile It to a .CMD Binary File.

For a LIMITED time
save over 65%...
This sale will not be
repeated after it's
over! *

SALE SPECIAL:

\$69.95

SPECIAL

Thank-You-Sale

Only From:

S.E. Media™
CPI
5900 Cassandra Smith Rd.
Hixson, Tn 37343
Telephone 615 842-6809
Telex 510 600-6630

A Division of Computer Publishing Inc.
Over 1,200 Titles - 6800-6809-68000

* K-BASIC will run under 6800X SK*DOS in emulation mode for the 6809.

Price subject to change without notice.

PT-68000 SINGLE BOARD COMPUTER

The PT68K2 is Available in a Variety of Formats
From Basic Kits to Completely Assembled Systems

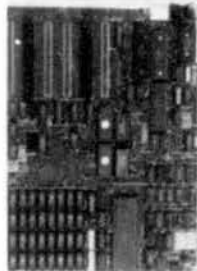
BASIC KIT (8 MHZ) - Board, 68000,
HUMBUG MONITOR + BASIC in ROM,
4K STATIC RAM, 2 SERIAL PORTS, all
Components \$200

PACKAGE DEAL - Complete Kit with
Board 68000 10 MHZ, SK'DOS, 512K
RAM, and all Necessary Parts \$575

ASSEMBLED BOARD (12 MHZ)
Completely Tested, 1024K RAM,
FLOPPY CONTROLLER, PIA, SK'DOS
\$899

ASSEMBLED SYSTEM - 10 MHZ
BOARD, CABINET POWER SUPPLY,
MONITOR + KEYBOARD, 80 TRACK
FLOPPY DRIVE, CABLES \$1299
For A 20 MEG DRIVE, CONTROLLER
and CABLES Add \$295

PROFESSIONAL OS9 \$500



FEATURES

- MC68000 Processor, 8 MHZ Clock (optional 10, 12.5 MHZ)
- 512K or 1024K of DRAM (no wait states)
- 4K of SPAM (6116)
- 32K, 64K or 128K of EPROM
- Four RS-232 Serial Ports
- Floppy disk controller will control up to four 5 1/4", 40 or 80 track.
- Clock with on-board battery.
- 2 - 8 bit Parallel Ports
- Board can be mounted in an IBM type PC/XT cabinet and has a power connector to match the IBM type power supply.
- Expansion ports - 6 IBM PC/XT compatible I/O ports. The HUMBUG™ monitor supports monochrome and/or color adaptor cards and Western Digital winchester interface cards.

PERIPHERAL TECHNOLOGY

1710 Cumberland Point Dr., Suite 8
Marietta, Georgia 30067
404/984-0742

VISA/MASTERCARD/CHECK/C.O.D.

Send For Catalogue

For Complete Information On All Products

™SK'DOS is a Trademark of
STAR-K SOFTWARE SYSTEMS CORP.
™OS9 is a Trademark of Microware

DATA-COMP

SPECIAL

Heavy Duty Power Supplies



For A limited time our HEAVY DUTY SWITCHING POWER SUPPLY. These are BRAND NEW units. Note that these prices are less than 1/4 the normal price for these high quality units.

Make: Boschert

Size: 10.5 x 5 x 2.5 inches

Including heavy mounting bracket and heatsink

Rating: in 110/220 volts ac (strap change) Out: 130 watts

Output: +5v - 10 amps

+12v - 4.0 amps

+12v - 2.0 amps

-12v - 0.5 amps

Mating Connector: Terminal strip

Load Reaction: Automatic short circuit recovery

SPECIAL: \$59.95 each

2 or more \$49.95 each

Add: \$7.50 each S/H

Make: Boschert

Size: 10.75 x 6.2 x 2.25 inches

Rating: 110/220 ac (strap change) Out: 81 watts

Outputs: +5v - 8.0 amps

+12v - 2.4 amps

+12v - 2.4 amps

+12v - 2.1 amps

-12v - 0.4 amps

Mating Connector: Molex

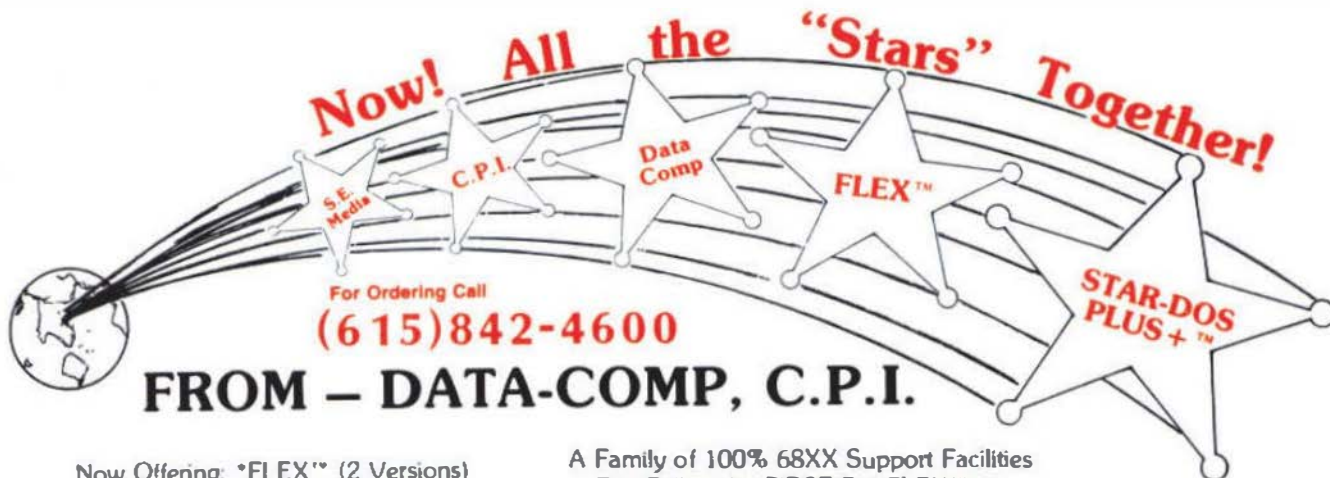
Load Reaction: Automatic short circuit recovery

SPECIAL: \$49.95 each

2 or more \$39.95 each

Add: \$7.50 S/H each

5900 Cassandra Smith Rd., Houston, Tx. 77343 Telephone 615 842-4600 Telex 510 600-6630



Now Offering: *FLEX* (2 Versions)
AND *STAR-DOS PLUS+™

A Family of 100% 68XX Support Facilities
The Folks who FIRST Put FLEX™ on
The CoCo

FLEX-CoCo Sr.
with TSC Editor
TSC Assembler

Complete with Manuals
Reg. \$250.⁰⁰ **Only \$79.⁰⁰**

STAR-DOS PLUS+

- Functions Same as FLEX
- Reads - writes FLEX Disks **\$34.⁰⁰**
- Run FLEX Programs
- Just type: Run "STAR-DOS"
- Over 300 utilities & programs to choose from.

FLEX-CoCo Jr.
without TSC
Editor & Assembler
\$49.⁰⁰

PLUS

ALL VERSIONS OF FLEX & STAR-DOS INCLUDE

TSC Editor
Reg \$50.00

NOW \$35.00

- + Read-Write-Dir RS Disk
- + Run RS Basic from Both
- + More Free Utilities

- + External Terminal Program
- + Test Disk Program
- + Disk Examine & Repair Program
- + Memory Examine Program
- + Many Many More!!!

TSC Assembler
Reg \$50.00

NOW \$35.00

CoCo Disk Drive Systems

2 THINLINE DOUBLE SIDED DOUBLE DENSITY DISK DRIVES
SYSTEM WITH POWER SUPPLY, CABINET, DISK DRIVE CABLE, J&M
NEW DISK CONTROLLER JPD-CP WITH J-DOS, RS-DOS OPERATING
SYSTEMS. **\$469.95**

* Specify What CONTROLLER You Want J&M, or RADIO SHACK

THINLINE DOUBLE SIDED
DOUBLE DENSITY 40 TRACKS **\$129.95**

Verbatim Diskettes

Single Sided Double Density **\$ 24.00**
Double Sided Double Density **\$ 24.00**

Controllers

J&M JPD-CP WITH J-DOS **\$139.95**
WITH J-DOS, RS-DOS **\$159.95**
RADIO SHACK 1.1 **\$134.95**

RADIO SHACK Disk CONTROLLER 1.1 **\$134.95**

Disk Drive Cables

Cable for One Drive **\$ 19.95**
Cable for Two Drives **\$ 24.95**

MISC

64K UPGRADE **\$ 29.95**
FOR C,D,E,F, AND COCO 11
RADIO SHACK BASIC 1.2 **\$ 24.95**
RADIO SHACK DISK BASIC 1.1 **\$ 24.95**

DISK DRIVE CABINET FOR A
SINGLE DRIVE **\$ 49.95**
DISK DRIVE CABINET FOR TWO
THINLINE DRIVES **\$ 69.95**

PRINTERS

EPSON LX-80 **\$289.95**
EPSON MX-70 **\$125.95**
EPSON MX-100 **\$495.95**

ACCESSORIES FOR EPSON

8148 2K SERIAL BOARD **\$ 89.95**
8149 32K EXPAND TO 128K **\$169.95**
EPSON MX-RX-80 RIBBONS **\$ 7.95**
EPSON LX-80 RIBBONS **\$ 5.95**
TRACTOR UNITS FOR LX-80 **\$ 39.95**
CABLES & OTHER INTERFACES
CALL FOR PRICING

DATA-COMP

5900 Cassandra Smith Rd.
Hixson, TN 37343



SHIPPING
USA ADD 2%
FOREIGN ADD 5%
MIN. \$2.50

(615)842-4600
For Ordering
Telex 5108008630

An Ace of a System in Spades! The New

MUSTANG-08/A™

Now with 4 serial ports standard & speed increase to 12 Mhz CPU + on board battery backup and includes the PROFESSIONAL OS-9 package - including the \$500.00 OS-9 C compiler! This offer won't last forever!

NOT 128K, NOT 512K FULL 768K No Wait RAM

The MUSTANG-08™ system took every hand from all other 68008 systems we tested, running OS-9 68K!

The MUSTANG-08 includes OS9-68K™ and/or Peter Stark's SKDOS™. SKDOS is a single user, single tasking system that takes up where "FLEX" left off. SKDOS is actually a 68XXX FLEX type system (Not a TSC product.)

The OS-9 68K system is a full blown multi-user, multi-tasking 68XXX system. All the popular 68000 OS-9 software runs. It is a speed whiz on disk I/O. Fact is the MUSTANG-08 is faster on disk access than some other 68XXX systems are on memory cache access. Now, that is fast! And that is just a small part of the story! See benchmarks.

System includes OS-9 68K or SKDOS - Your Choice Specifications:

CPU	MC68008	12 Mhz
RAM	768K	256K Chips
	No Wait States	
PORTS	4 - RS232	MC88B1 QUART
	2 - 8 bit Parallel	MC8821 PIA
CLOCK	MC48T02	Real Time Clock Bat. B/U
EPROM	16K, 32K or 64K	Selectable
FLOPPY	WD1772	5 1/4 Drives
HARD DISK	Interface Port	WD1002 Board

Now more serial ports - faster CPU
Battery B/U - and \$850.00 OS-9 Profes-
sional with C compiler included!

*\$400.00

See Mustang-02 Ad - page 5
for trade-in details



MUSTANG-08

LOOK

Seconds 32 bit Register
Integer Long

Other 68008 8 Mhz OS-9 68K...18.0...9.0

MUSTANG-08 10 Mhz OS-9 68K...9.8...6.3

Minis()

C Benchmark Loop

```
/* Init I; */
register long I;
for (I=0; I < 999999; ++I);
```

Now even faster!
with 12 Mhz CPU

C Compile times: OS-9 68K Hard Disk	
MUSTANG-08 8 Mhz CPU	0 min - 32 sec
Other popular 68008 system	1 min - 05 sec
MUSTANG-020	0 min - 21 sec



25 Megabyte
Hard Disk System

\$2,398.90

Complete with PROFESSIONAL OS-9
includes the \$500.00 C compiler, PC
style cabinet, heavy duty power supply,
5" DDDS 80 track floppy, 25 MegByte
Hard Disk - Ready to Run

Unlike other 68008 systems there are several significant differences. The MUSTANG-08 is a full 12 Megahertz system. The RAM uses NO wait states, this means full bore MUSTANG type performance.

Also, allowing for addressable ROM/PROM the RAM is the maximum allowed for a 68008. The 68008 can only address a total of 1 Megabytes of RAM. The design allows all the RAM space (for all practical purposes) to be utilized. What is not available to the user is required and reserved for the system.

A RAM disk of 480K can be easily configured, leaving 288K free for program/system RAM space. The RAM DISK can be configured to any size your application requires (system must have 128K in addition to its other requirements). Leaving the remainder of the original 768K for program use. Sufficient source included (drivers, etc.)

FLEX is a trademark of TSC

MUSTANG-08 is a trademark of CPI

Data-Comp Division



A Decade of Quality Service*

Systems World-Wide

Computer Publishing, Inc. 5900 Cassandra Smith Road
Telephone 615 842-4601 - Telex 510 600-6630 Hixson, TN 37343

* Those with SWTPC hierarchy FLEX 5 - Call for special info.